VM ALLOCATION IN CLOUD DATACENTERS BASED ON THE MULTI-AGENT SYSTEM

ASHRAF M. S. AL-OU'N

PhD

UNIVERSITY OF BRADFORD

2017

VM Allocation in Cloud Datacenters Based on the Multi-Agent System

An Investigation into the Design and Response Time Analysis of a Multi-Agent-based Virtual Machine (VM) Allocation/Placement Policy in Cloud Datacenters

Ashraf Mohammad Salman AL-OU'N

Submitted for the degree of

Doctor of Philosophy

Department of Computer Science

School of Electrical Engineering and Computer Science

Faculty of Engineering and Informatics

University of Bradford

2017

Abstract

Ashraf Mohammad Salman Al-Ou'n

VM Allocation in Cloud Datacenters Based on the Multi-Agent System

An Investigation into the Design and Response Time Analysis of a Multi-Agent-based Virtual Machine (VM) Allocation/Placement Policy in Cloud Datacenters

Keywords: Virtual Machine (VM); Allocation; Response Time; Agent-based; Cloud Computing; Datacentre; Virtualization; Multi-Agent System; Host; Physical Machine (PM); Processing Element (PE).

Recent years have witnessed a surge in demand for infrastructure and services to cover high demands on processing big chunks of data and applications resulting in a mega Cloud Datacenter. A datacenter is of high complexity with increasing difficulties to identify, allocate efficiently and fast an appropriate host for the requested virtual machine (VM). Establishing a good awareness of all datacenter's resources enables the allocation "placement" policies to make the best decision in reducing the time that is needed to allocate and create the VM(s) at the appropriate host(s). However, current algorithms and policies of placement "allocation" do not focus efficiently on awareness of the resources of the datacenter, and moreover, they are based on conventional static techniques. Which are adversely impacting on the allocation progress of the policies.

This thesis proposes a new Agent-based allocation/placement policy that employs some of the Multi-Agent system features to get a good awareness of Cloud Datacenter resources and also provide an efficient allocation decision for the requested VMs. Specifically, (a) The Multi-Agent concept is used as a part of the placement policy (b) A Contract Net Protocol is devised to establish good awareness and (c) A verification process is developed to fully dimensional VM specifications during allocation. These new results show a reduction in response time of VM allocation and the usage improvement of occupied resources.

The proposed Agent-based policy was implemented using the CloudSim toolkit and consequently was compared, based on a series of typical numerical experiments, with the toolkit's default policy. The comparative study was carried out in terms of the time duration of VM allocation and other aspects such as the number of available VM types and the amount of occupied resources. Moreover, a two-stage comparative study was introduced through this thesis. Firstly, the proposed policy is compared with four state of the art algorithms, namely the Random algorithm and three one-dimensional Bin-Packing algorithms. Secondly, the three Bin-Packing algorithms were enhanced to have a two-dimensional verification structure and were compared against the proposed new algorithm of the Agent-based policy. Following a rigorous comparative study, it was shown that, through the typical numerical experiments of all stages, the proposed new Agent-based policy had superior performance in terms of the allocation times. Finally, avenues arising from this thesis are included.

Declaration

I hereby declare that this thesis has been genuinely carried out by myself and has not been used in any previous application for a degree. Chapters 4 to 7 describe work performed in conjunction with my supervisors, Prof. D. D. Kouvatsos and Dr. Mariam Kiran. The Chapters (4 and 5) have been accepted for publication as a conference paper in 2015. The invaluable participation of others in this thesis has been acknowledged where appropriate.

Ashraf Mohammad Al-Ou'n.

Dedication

This thesis is dedicated to my beloved parents, whose love, help, support and prayers are the reason why I am where I am today, my life partner and my beloved wife, my beloved children (Mohammad, Osama and Fatima), my beloved nephew (Ekrema), and my beloved sisters and brothers.

Acknowledgements

In the Name of Allah, the Most Gracious, the Most Merciful. All Praise is Due to Allah for His Glorious Ability and Great Power. I would like to thank Allah for giving me the power, patience and knowledge to complete this doctoral thesis.

I would like to express my gratitude to all those who gave me the support and encouragement to complete this thesis. First of all, I would like to express my sincere appreciation and profound gratitude to my supervisors Professor Demetres Kouvatsos and Dr. Mariam Kiran, who's provided invaluable guidance throughout the course of this research. I would like to express my appreciation for their helpful suggestions, continual and unwavering encouragements, patience, kindness and endless support throughout the entire research and thesis writing process.

I would like to express my appreciation to all my colleagues and friends for their fruitful discussions, suggestions, co-operation and continuous support.

I am deeply indebted to my family, relatives and friends who offered me great support and encouragement throughout my studies. My heartfelt thanks go to my parents, my wife, my sons, my nephew, my sisters and my brothers for their unfailing love, prayers continuous support and patience.

Finally, I would like to gratefully acknowledge the provision of the scholarship from the Al albayt University in Jordan.

List of Contents

List of Tables

List of Algorithms

List of Figures

List of Acronyms

1D	**One-Dimensional**
2D	**Two-Dimensional**
Ack	**Acknowledgement or Acknowledge**
ACL	**Agent Communication Language**
AI	**Artificial Intelligence**
AWS	**Amazon Web Services**
BD	**Big-Data**
BF	**Best-Fit algorithm (Bin-Packing)**
BFD	**Best-Fit Decreasing algorithm (Bin-Packing)**
BW	**Bandwidth or System Bus Bandwidth**
Cloudlet	**Cloud computing Application/Task**
CPU	**Central Processing Unit / Processor**
DC	**Data Centre or Datacenter**
FIPA	**The Foundation for Intelligent Physical Agent**
FT	**First-Fit algorithm (Bin-Packing)**
FTD	**First-Fit Decreasing algorithm (Bin-Packing)**
Host	**Cloud computing Server or Physical Machine**
HPC	**High-Performance Computing**
HPG	**Highest Potential Growth**
KQML	**The Knowledge Query and Manipulation Language**
MAS	**Multi-Agent System**
MBFD	**Modification of Best-Fit Decreasing algorithm (Bin-Packing)**

MIPS	**Millions of Instructions Per Second**
MM	**Minimization of Migration**
OMNet++	**Objective Modular Network Testbed in C++ (toolkit)**
OPNet	**Optimized Network Engineering Tools (toolkit)**
PABFD	**Power Aware Best-Fit Decreasing algorithm**
PE	**Processing Element or Processor**
PM	**Physical Machine (Server)**
QoS	**Quality of Service**
QoSV	**Quality of Service Violations**
SLA	**Service Level Agreement**
ST	**Single Threshold**
VM	**Virtual Machine**
VMM	**Virtual Machine Monitor (Virtual Layer/Hypervisor)**
WF/MF	**Worst-Fit/Max-rest-Fit algorithm (Bin-Packing)**
WFD/MFD	**Worst-Fit/Max-rest-Fit Decreasing algorithm (Bin-Packing)**
Xen	**Type of VMM/Hypervisor**

Chapter 1 Introduction

1.1 Introduction

Recent advances in high-performance computing, social networking and big-data processing, has led to high demands for additional computational power in computer networks supported by cloud computing platforms. In this context, it is vital importance to provide a cloud datacenter with a large-scale infrastructure to efficiently support applications and service requirements [1-3].

The employment of a virtualization technology in the cloud datacenter takes place along with capacity extensions and an increase of the number of servers; wherein the cloud datacenter provides a virtualization layer, a virtual machine monitor (VMM) through servers (physical machines) to support the virtual machines. This means increasing utilization of the resources and the complexity inside the datacenter because one server (Host) can serve more than one user through a virtual machine (VM) [4]. Thus, it is highly required to constantly design new allocation policies towards the efficient handling of high complexity and capacity of the cloud datacenters in order to improve the allocation time of occupied resources.

Computational resources are composed of multi-thousands of physical machines (PMs) inside a datacenter to host the VMs that process the end user's and client's requests [5, 6]. Therefore, the VM allocation policies are very important to find and allocate or "place" a VM with an appropriate Host (PM: physical machine) to VM requirements. The VM allocation policies can be classified into two main categories: i) The first type provisions and places VMs on the proper hosts according to requests received from brokers and end users, and ii) The second type optimizes the current allocation of VMs [7, 8].

Note that, most of current algorithms for placement 'allocation' policies belong to the second category of allocation 'placement' policies [6, 7], which focuses on the optimization of placement 'allocation' during the VM's lifecycle e.g. policies in [9-12] for a particular purpose such as an energy efficient and SLA(Service Level Agreement). Moreover, the current state of the art policies are depending on traditional (conventional) techniques to perform the allocation process such as the Bin-Packing algorithms [13], further, the allocation time of allocation 'placement' has not been one of the aspects of policies evaluation.

To this end, it is highly important to improve the design and development of efficient methodologies and implementation techniques that can allow and adjust the dynamic behaviors of cloud computing systems. This thesis proposes the use of multi-agent technology for large-scale datacenters in order to search, allocate and update the applications, resources (i.e. VMs) and massive volumes of data. In this way, energy efficient cloud computing infrastructures will be supported with intelligent allocation policies, provisioning algorithms, monitoring services and data access services [14].

The proposed new Agent-based VM allocation policy aims to improve the allocation "placement" progressing based on the Multi-Agent system technology using the CloudSim toolkit [15, 16]. This policy adopts the Contract Net Protocol as a communication approach between agents through the Multi-Agent system to facilitate the establishment of good awareness over the Datacenter resources pool. It also verifies the full-dimensional of VM's specifications. In this thesis, the allocation time is considered as the main measurement for verifying the policies performance and efficiency during allocation 'placement' progress. In addition, the Agent-based VM placement policy makes use of other measures such as the number of available VM types and the amount of occupied resources of VM allocation process among the entire Hosts of the cloud datacenter.

The Agent-based VM allocation policy is implemented according to the configuration of CloudSim toolkit; the multi-agent system of the proposed VM allocation policy depends on two simple make-decision agents: (i) the Datacenter_Coordinator agent; and (ii) the Host_Agent (inter-leaf agent); one for each host in the datacenter. Further, the Contract Net Protocol is used as a communication and coordination approach between the Datacenter_Coordinator and the Host_Agent's; to identify and find the proper host to the requested VM, then the Datacenter_Coordinator selecting one of the passed hosts according to the First-Pass-Fit mechanism.

Note that, the Agent-based policy through the host verifying/testing process, covers all the VM specifications/requirements; this means the Agent-based policy is a full-dimensional VM placement/allocation policy. Additional, in order to prove the concept of using the multi-agent system among the cloud datacenter for the VM allocation; a comparison is made between the Agent-based policy and the default VM allocation policy of CloudSim toolkit

over six different scenarios, which are divided equally into two categories: i) Time-Shared scheduler scenarios [17] and secondly, Space-Shared scheduler scenarios [18].

In order to prove the potential of the proposed allocation policy, two comparative studies are introduced, namely the proposed Agent-based policy and some of the state of the art allocation/placement algorithms/policies. Through the first comparative study six scenarios are presented between the Random algorithm [7, 9], three one-dimensional Bin-Packing algorithms (First-Fit, Worst-Fit and Best-Fit [13]) and the Agent-based policy [19], where the four state of the art algorithms have been implemented and adjusted according to the CloudSim toolkit configurations. In the two-dimensional comparative study, just the Bin-Packing algorithms and the Agent-based are tested through the numerical experiments. The development of the verification criteria in the Bin-Packing algorithms from one-dimensional to a two-dimensional, this necessitates the implementation of two versions for each algorithm (six algorithms in total), due to their static nature (static mechanism) and because the CloudSim toolkit supports two VM schedulers (Time-Shared, Space-Shared) on the host level. It will be shown in this thesis that, the Agent-based policy showed efficient performance and highly dynamic nature through the two comparative studies, despite of keeping the same structure and functionality without any modification and improvement.

1.2 Motivation

The cloud computing is an emerging alternative field to that of Distributed computing in the art stage as a computing service provider [20]. Moreover, it has dynamic essential characteristics such as Rapid Elasticity [20] and Resource Pooling [20], which make the provisioning and releasing of resources and services automatically without needing to human intervention [20]. Thus, in the cloud datacenters, the design and implementation of modern efficient methodologies, algorithms and policies for allocation, provisioning and scheduling of the resources and services has high significance for the tracking, handling and adjusting of the dynamic behavior of cloud computing [5, 9, 15, 20].

Furthermore, the huge expansion in the infrastructure of the cloud datacenters during the recent years, such as Google datacenters [21] and Microsoft datacenters [22] supports the high demands of computational power for the cloud services and applications [1-3]. In addition, the broad use of virtualization technology through the datacenter hosts and the infrastructure

expansions make the searching to identify the proper host to the requested VM through the allocation process very complex. Consequently, this slows down the cloud datacenter system response to the VMs queries [4-6].

The significance of using the multi-agent system as a non-conventional and intelligent technique through the allocation process, relates to the resulting reduction of the response time of the datacenter and mitigate the impact of complexity [14]. This is particularly important as, current VM allocation/placement algorithms/policies are conventional or based on traditional techniques like the Bin-Packing algorithms (i.e. First-Fit and Best-Fit [13]).

This thesis focuses on the first VM allocation category (allocation/placement) and depends on the allocation time as an evaluation measurement/standard to the VM allocation algorithm/policy performance and functionality. Note that the majority research works in VM allocation algorithms/policies belong to the second category of the VM allocation, which is concerned with the optimization of current allocated VMs in the datacenter according to special reasons such as the Power consumption [6-8].

1.3 Aims and Objectives

The main aims of the presented research in this thesis is i) To apply the multi-agent system technology on the VM allocation process of the cloud datacenter. ii) To establish a good awareness of datacenter resources (resource pooling). And iii) To improve the performance of the VM allocation policy/algorithm during the identification process to the proper host and the creation process to the requested VM in the datacenter.

These main objectives of the thesis are as follows.

1. To propose an efficiently conceptual structure of the multi-agent system, which is suitable and adaptable to the structure and configurations of the cloud computing datacenter environment.
2. To use a suitable Cloud simulation toolkit, as a cloud computing environment simulator, to implement and evaluate the multi-agent system and the VM allocation policy.
3. To implement a simple multi-agent system based on the datacenter structure for using through the VM allocation process among datacenter's hosts.

4. To use a suitable communication and coordination approach between the agents in the multi-agent system; to improve the cooperation and avoid conflicts between system agents.
5. To design, implement and assess a new VM allocation policy depending on the multi-agent system; to improve the allocation progress and mitigate the impact of the searching complexity.
6. To use some of the state of the art VM allocation algorithms, in order to evaluate the concept of using the multi-agent system in the VM allocation process, based on suitable measures of performance. Moreover, to construct a comparative study with the proposed Agent-based VM allocation policy.

1.4 Contributions

This work carries out an investigation study to the VM allocation process through the cloud computing datacenter. The study consists of many research components, where all the research stages are compatible and adjustable with the cloud datacenter structure and design according to cloud computing environment configurations. Moreover, the virtualization technology is employed for the cloud computing in a dynamic manner over datacenter infrastructure.

In this thesis a new Multi-Agent system is employed through the large-scale virtualized cloud computing datacenters. For improving the efficiency of provisioning, placement, updating and scheduling policies among the datacenter's resources (hosts). More specifically, apply a Multi-Agent system technology in VM allocation process by proposing and implementing a new Agent-based VM allocation policy for virtualized datacenters.

By addressing the research objectives and research aims, this thesis makes the following contributions.

1. Two computing fields have been combined during this research work, whereas the multi-agent system (c.f., Artificial Intelligent (AI field)) used for the first time as a part of our proposed solution to the VM allocation process in cloud datacenters (Networks field).

2. A conceptual structure and design to the multi-agent system has been proposed, which is compatible and adjustable to the cloud datacenter structure and design; this means that the multi-agent system can be very useful to the functionality and performance of the provisioning and scheduling algorithms within the datacenter infrastructure, such as the VM allocation process.

3. Design, implementation and verification of new proposed Agent-based VM allocation policy for large-scale virtualized datacenters. Practically, the new Agent-based policy is considered as a non-conventional VM allocation policy. Because it's implementation based on a special Multi-Agent system, which is developed according to the structure and design of virtualized datacenter. Moreover, using an effective communication and coordination approach between the agents to construct a good awareness about the datacenter's resources (hosts).

4. The introduction of full-dimensional verification criteria and First-Pass-Fit selection criteria to the proposed policy. In this context, the verification criteria of the policy consider all the VM requirements through verifying the host in the allocation process and consequently, choose the first passed host from the verification stage.

5. A rigorous comparative study to verify and testify the potential of Agent-based policy compared with the state of the art algorithms. Specifically, the comparative study consisted of two stages based on the structure and the implementation of the state of the art algorithms (one-dimensional, two-dimensional).

6. The use of the allocation time and the amount occupied resources (i.e. created VMs in hosts) through the datacenter as measurement standards, to evaluate and assess the performance and functionality of the VM allocation policies through numerical experiments. Moreover, the balanced distribution of VMs types and the flexibility of VM allocation algorithms are used in some scenarios as measurement standards for verifying the VM allocation policies performance.

1.5 Thesis Organization

Chapter 2 introduces some relevant research works on the state of the art in Sections 2.2 and 2.3. Section 2.2 focuses on the allocation and placement algorithms/policies of VM through the cloud datacenter, which are simulated by using the CloudSim toolkit. One of them is the Modification of Best-Fit Decreasing (MBFD)/Power Aware Best-Fit Decreasing (PABFD) algorithm [7-12], which was used as a part in some research works relating to Energy-Awareness with Service-Level Agreement (SLA). Section 2.3 presents another research work relating to the HPC Cluster Trace and Synthetic Generated workloads without using the CloudSim toolkit, which aims to prevent any SLA violations and unnecessary VM migration.

Chapter 3 introduces a full review for the CloudSim toolkit and its components; this is because the CloudSim toolkit represents the cloud computing environment and it is the platform of simulation and evaluating the proposed VM allocation policy (Agent-based policy). Moreover, this chapter either answers some important questions such as 'why we choose the simulation instead of Test-Bed? What the novelty of CloudSim toolkit especially for allocation and provisioning resources algorithms/policies?' or, defines some important standards and scope (work-space) such as the scope of allocation/placement policies, allocation time, VM lifecycle, allocation dimensions and VM schedulers especially on the host level.

Chapter 4 states the assumptions and propositions for the proposed conceptual structure and design of the multi-agent system as applied to the cloud datacenter in conjunction with the design, implementation, structure, testing of the proposed policy and associated selection criteria. It also describes the importance of designing and developing new provision and allocation policies in the cloud datacenter. Moreover, the design and implementation of the new Agent-based policy through the CloudSim toolkit is presented and the use of the Contact-Net Protocol as a coordinating approach between Agents through the new policy is explained.

Chapter 5 defines the mathematical definition of the model for some significant standards, namely the *Allocation Time*, *Cumulative Allocation* and *Turnaround Times,* according to the CloudSim toolkit in order to validate the potential of the proposed new Agent-based policy. The latter is based on the use of the Multi-Agent system [19]. Furthermore, it presents general fixed conditions and assumptions for all scenarios in the context of this thesis. Finally, it

includes the results of the numerical experiments of this research by means of 6 scenarios in order to compare the performance of the proposed new Agent-based VM allocation policy versus that of the default VM allocation policy of CloudSim toolkit.

Chapter 6 highlights the importance of the comparative study for the new proposed Agent-based policy versus the state of the art solution and techniques. In particular, it introduces a one-dimensional comparative study between the new Agent-based policy versus four the state of the art policies over six different scenarios. Consequently, the implementation and development of the Random algorithm and three one-dimensional Bin-Packing algorithms are used for the comparisons experiments.

Chapter 7 carries out the second stage of the comparative study between the new Agent-based policy and three two-dimensional Bin-Packing algorithms under Time-Shared scheduler configurations and implements a comparative study through two scenarios. Moreover, it includes the development of the three two-dimensional Bin-Packing algorithms under the Space-Shared scheduler and performs the comparative study through two earlier used scenarios. Finally, it employs a special scenario in order to assess the flexibility and dynamic nature of all six policies in this chapter.

Finally, **Chapter 8** summaries the conclusions of this thesis and makes recommendations for future works.

Chapter 2 Related Works

2.1 Introduction

This chapter introduces a brief illustration to some related research works to the context of this thesis, which is the Agent-based VM allocation/placement algorithm. Hence, the VM allocation policies/algorithms can be classified into two main categories: the first category is to provision and place VMs on the proper hosts, according to requests received from brokers and end users. The second category is to optimize (reallocation) the current allocation of VMs based on defined conditions such as, CPU utilization, power consumption, etc... [7, 8].

However, the context of this thesis is evaluated through the CloudSim toolkit c.f. Chapter 3, due to that the chapter focuses on the related works, which are evaluated through the same cloud computing simulation toolkit.

2.2 CloudSim Toolkit Related Works

This section introduces some of the related works of the thesis context (VMs allocation/placement algorithms), which used the CloudSim toolkit as a cloud computing environment to imitate and evaluate the potential and improving of these algorithms/policies over VMs allocation/placement process in the cloud computing datacenter.

2.2.1 Modification of Best-Fit Decreasing (MBFD)/ Power Aware Best-Fit Decreasing (PABFD)

The authors/researchers of [7-12], have used a modification of the Best Fit Decreasing (BFD) algorithm for VM allocation, through datacenter resource management approaches, VMs optimization/consolidation policies and cloud computing architectural vision. Where, all these approaches, policies and architectural vision associate in common objective, which is the cloud computing datacenter energy-efficiency with high QoS. This means, all research works tried to reduce the power consumption in datacenters without the quality of service violations (QoSV); based on datacenter energy-awareness with the Service-level Agreement (SLA).

In [12], the authors renamed MBFD algorithm by Power Aware Best-Fit Decreasing (PABFD), but the mechanism of both algorithms almost same without modification. The following two points illustrate the mechanism of MBFD algorithm:

- Sorting all VMs in decreasing order of current power utilization.
- Allocate each VM to a host, which provides the least increase of power consumption according to the current allocation.

Furthermore, all the energy-efficient approaches/policies aim to reduce the power consumption along with preserving the user QoS, through switch-off all idle hosts/servers depends on dynamic VMs reallocation/optimization according to the live VM migration concept. However, the VMs reallocation/optimization policies consist of two stages:

1. VMs selection: through this stage, the VMs are selected to migrate according to some of the heuristics selection algorithms such as; a Single Threshold (ST), Minimization of Migration (MM) and Highest Potential Growth (HPG). Additional, all selection algorithms can be classified according to the number of using thresholds like, single-threshold and two-thresholds.
2. VMs placement: again, the MBFD algorithm is used to place the selected VMs through the first stage in the proper hosts.

Further work in [10-12] was presented using a decentralized resource management in the datacenter, which for assisting to achieve the same purposes; based on a software architecture model. Additionally, the software architectural model in [11, 12] consists of two layers:

- Local Manager, which resides in the VMM of each node/host; for monitoring the CPU utilization, resizing the VM according to the resources needed, and deciding when and which VMs should be migrated from the node/host.
- Global Manager, which resides in the master node (datacenter administrator); to collect the information from the local managers, and issue the commands for the optimization of the VMs placement.

Moreover, in [10] the authors added a Dispatcher (extra layer) to the software architecture model; which dispatches the requested VMs to multi-global managers to place/allocate them among the datacenter nodes/hosts. In this context, the local manager depends on another monitoring factors, along to the CPU utilization; which are virtual network topologies

established between the VMs (communication state), and the thermal state of the computing node/host.

2.2.2 VMs Allocation Policy for Data-Intensive Applications

In [6], a special policy/approach for VM allocation "placement" was presented for data-intensive applications, where, the data-intensive applications need to access and communicate with the data frequently. Thus, the logical/physical distance, the network I/O performance and the network instability between the VMs and data storage; might affect significantly on the application progressing.

The approach aims to minimize the data-transfer time consumption between the VMs (host/PM) and data access storage, through two stages: (i) VMs placement, and (ii) VMs migration (VMs optimization) based on the data-transfer time. Therefore, firstly the placement algorithm places the requested VMs in hosts/PMs with shortest data-transfer time, then if the data-transfer time exceeds a determined threshold the migration algorithm is triggered to optimize the VM allocation.

In this context, the data-transfer time depends on (a) the logical/physical distances between the host/PM who hosts the VM and data access storage, (b) network conditions between the PM/host and the data storage.

2.2.3 Multi-objective VM Placement

Through [23], a multi-objective VM placement policy/approach is presented; where the VM placement problem was formulated as a multi-objective optimization for minimizing the following management aspects: (a) total resource wastage, (b) power consumption, (c) thermal dissipation costs. Consequently, the authors proposed a two-level control system, which uses an improved/modified genetic algorithm with fuzzy multi-objective evaluation; for efficiently searching the large space (a huge datacenter) and suitably combining possible conflicting objectives.

However, the two-level control system consists of Local-Controller and Global-Controller like [10-12] c.f. Section 2.2.1. The following points illustrate the controllers:

- Local-Controller: which determines the resource amount needed by an application, and it resides in the virtual layer (VMM) of the local node/host.
- Global-Controller: which receives the request of VM placement or migration, information about power model and temperature model. Also, it determines the VM placement and resource allocation.

Practically, this approach for optimizing the VM placement problem has two stages:

- Searching: use a modified genetic algorithm for efficiently searching for the global optimal solutions to the new VM placement or migration requests.
- Combining & Evolution: use a Fuzzy-logic based approach for evaluating the combination of different objectives together, then selecting the optimal combination.

Eventually, the work has been compared with four well-known Bin-Packing algorithms and two single-objective approach. According to their results and conclusions; this approach showed good balance among the conflicting objectives, but others cannot.

2.2.4 A Hybrid Energy-Aware VM Allocation Approach

Power-aware VM scheduling-policy was presented in [24], which aims to decrease the power consumption with least SLA violations; by minimizing the overall number of used servers and the VMs migrations. The hybrid VM allocation approach depends on two algorithms to achieve his objectives through two stages. The first stage uses a VM placement algorithm, which attempts to identify the proper host for the requested VMs based on a defined static CPU utilization threshold (upper-threshold), to prevent any SLA violations.

The second stage uses an optimization algorithm; which defines another threshold (lower-threshold) according to the current CPU utilization status of hosts/nodes, then using the VM live migration technique to move all VMs in hosts/nodes less than this threshold to new hosts/nodes. But, the new hosts/nodes must be less than the static CPU utilization threshold (upper one) after the migration of VMs. Finally, the idle hosts/nodes are switched-off to minimize the power consumption.

The next section presents one of the related works, which uses another technique to evaluate the work through cloud computing environment.

2.3 HPC Cluster Trace and Synthetic Generated Workloads.

In [25], a VM Deadline driven placement policy was presented as a part of semi-static VM consolidation approach, where the semi-static VM consolidation allows the migration if the host/server of VM is turned-off just only. However, this approach aims to save energy costs of the datacenter with least SLA violations by preventing (a) any SLA violations and (b) any unnecessary VM migration.

Specifically, it contests of two stages (algorithms): the first stage, pre-determines the turn on/off hosts/PMs across a day, according to the prediction of workload variations based on historical measurements. In the Second stage, a VM Deadline driven placement algorithm is used; to place the VMs in the proper hosts/PMs based on the hosts/PMs turn on/off schedule from the first stage (algorithm). Additionally, the VM scheduling/placement decision depends on:

- Prevent SLA violation: place the VM in host/PM, which has enough resources during the execution time of arriving job/application (VM lifecycle).
- Prevent VM migration: place the VM in host/PM that might remain power-on throughout the job/application execution time (VM lifecycle).

Eventually, this consolidation approach is evaluated by using a real HPC cluster trace and synthetic generated workloads. Also, the results are compared with three Bin-Packing algorithms (First-Fit, Best-Fit and Worst-Fit) and Random algorithm.

2.4 Discussion

Most of the presented works through this chapter focuses on VM optimization and consolidation according to the stated objectives such as a minimizing the power consumption and reducing the SLA violations. Where, most of the current research works in VM allocations belong to the second category of VM allocation policies, which is the optimization one.

Earlier works in the field do not take into consideration the VM Allocation Time, amount of occupied resources (created VMs in hosts) in the datacenter. And, the Allocation process associated with various VMs types and their distribution as a standard to performance and functionality validation.

In addition, the current research efforts do not focus on improving the functionality and techniques of the VM allocation policies in the virtualized datacenter. To cope the dynamic manner of cloud computing and reduce the impact of high complexity over searching process through the virtualized datacenter's resources. Moreover, almost all VM allocation policies and VM optimization consultant approaches based on conventional algorithms such as, the Bin-Packing algorithms. Which produces a weakness to keep pace with dynamic characteristics of cloud computing as mentioned before, like a rapid elasticity and resources pooling [20].

Eventually, the current research works in VM allocation policies and optimization approaches do not have full awareness about firstly, all datacenter's resources (hosts). Secondly, the provisioning and scheduling policies through the hosts like, Time-Shared and Space-Shared schedulers. Moreover, most of current VM allocation policies do not offer full-dimensional verification criteria in dynamic style.

Note in this thesis the Agent-based VM allocation policy is proposed, which is based on the Multi-Agent system technology through the allocation process and it takes into consideration the *Allocation Time* as a standard or measurement to the VM allocation.

2.5 Summary

This chapter addressed related works of the literature on VM allocation/placement and optimization algorithms that used the CloudSim toolkit to evaluate their results since this toolkit is used through this thesis as a platform/environment to evaluate the Agent-based policy.

The next chapter presents a full illustration about the CloudSim toolkit as a simulator to the cloud computing environment.

Chapter 3 CloudSim Toolkit

3.1 Introduction

The configuration, composition and deployment requirements are different for some of the emerging and conventional cloud-based application services, such as web hosting, social networking, content delivery and real-time instrumented data processing [4, 15, 16]. So, in the real cloud computing environment (Google App Engine[26], Amazon EC2[27], AWS – Cloud Computing Services[28], Microsoft Azure[29, 30]); measuring the performance of allocation and scheduling policies (provisioning) for different application models is tremendously challenging under temporary conditions, due to: (i) cloud shows variety demands, system sizes, supply patterns and resources (network, hardware, software); (ii) divers users with divers, competing and dynamic QoS requirements; (iii) applications have various specifications such as a performance, workload and dynamic scaling requirements[4, 15, 16].

Using a real infrastructure like Microsoft Azure and Amazon AWS as benchmarking for the application performance (throughput and cost benefits) is usually constrained by the stiffness of the cloud infrastructure especially under mutable conditions such as availability and workload patterns. Which makes the repetition of the experiments to obtain an authenticated results extremely difficult mission. Moreover, the re-configuration of benchmarking parameters over multiple trials runs across a large-scale cloud computing infrastructure is time-consuming and tiresome. In this context, it is impossible to perform benchmarking trials in dependable, repeatable and scalable platforms (environments) using real-world Cloud platforms (environments); because the developers of application services have no full control over the cloud environments that causes such limitations [4, 15, 16].

The simulation tools come as a more applicable alternative of real-cloud benchmarking. The simulation approach opens up the possibility for anyone to evaluate the presumption of the real-world application benchmarking study within a fully controlled environment and easily obtain results. Hence, considerable benefits are offered by the simulation-based approaches to the researchers and the IT companies through: (i) testing the application services within repeatable and fully controlled atmosphere, (ii) adjusting the system bottlenecks before installing and releasing the services on real cloud computing environments, (iii) verifying the

services on simulated infrastructures with varying workload mix and different resource performance scenarios to experiment and develop adaptive application provisioning policies and techniques [4, 15, 16, 31].

The authors [4, 15, 16] present the CloudSim toolkit as a cloud modeling simulator, which permits smooth modeling, simulation and experimentation of cloud computing application services and environments. Because the current simulators of the distributed systems (Network, Grid) [4, 15, 16, 32-34] none of them can be used immediately for simulating and modeling the cloud computing [4, 15, 16].

3.2 The CloudSim Toolkit

CloudSim toolkit is an extensible simulation toolkit based on SimJava toolkit and GridSim toolkit; it provides modeling and simulation of Cloud computing systems and application provisioning environments. Moreover, both system and behavior modeling of Cloud system components such as virtual machines (VMs), datacenters, brokers and resource provisioning policies are supported by the CloudSim toolkit [4, 15, 16].

The CloudSim toolkit presents generic application provisioning techniques; that gives researchers an opportunity to extend them with limited effort and apply them to their particular works. Currently, the simulation and modeling of Cloud computing environments consisting of both single and inter-networked clouds (federation of clouds) are supported [4, 15, 16]. Furthermore, CloudSim toolkit presents custom interfaces for implementing provisions techniques and policies for allocation of VMs under inter-networked Cloud computing scenarios [4, 15, 16].

Specifically, CloudSim toolkit shows the following novel features [4, 15, 16, 35, 36]:

- Supports a simulation and modeling of large-scale Cloud computing environments, on a single physical computing node, including datacenters.
- Offers an autonomous platform for modeling whole cloud system component such as: Clouds, service brokers, provisioning, and allocation policies.
- Enables the simulation of network connections among the simulated system elements.

- Supports the simulation of a federated Cloud environment that inter-networks, resources from both public and private domains.

Moreover, the CloudSim toolkit offers some of the unique features that would speed up the development of new application provisioning algorithms of Cloud computing [4, 15, 16]:

- The virtualization engine, which aids to create and manage multiple, independent, and co-hosted virtualized services on a datacenter node.
- The ability and flexibility for switching between space-shared and time-shared allocation of processing cores to virtualized services.

This section introduced a novel features and characteristics of CloudSim toolkit, for simulating the cloud computing environment. which makes the CloudSim a favorable selection for simulating and implementation the proposed research work in the cloud computing field, in comparison with other current simulation toolkits[4, 15, 16], like OMNet++ and OPNet. In addition, CloudSim toolkit offers an efficient simulation of Cloud computing system components directly without needs to new design and implementation, such as a Datacenters, Brokers and allocation policies. In contradictory, the OMNet++, for instance, needs to design, model and implement of previous cloud computing components from scratch.

Moreover, CloudSim provides excellent tools for building and simulating the Virtualization layer (VMM and VM) in the Datacenter's hosts, which gives the CloudSim toolkit the usage privilege to verify any new proposed VM allocation policy. In contrast, most of the current simulation toolkits such as OMNet++ and OPNet do not provide the virtualization layer (VMM and VM) as a default component of the simulator. Note that, the virtualization layer is one of most important of cloud computing characteristics compared to the conventional service provider platforms like, a Grid computing.

Note that, based on the CloudSim toolkit novel features and characteristics that are mentioned in this section. It is chosen in this work for the verification and checking of the proposed algorithm for VM allocation "placement".

3.3 CloudSim Toolkit Classes Design

In this section, the classes' breakdown, general features and characteristics of the CloudSim toolkit are presented. That can be very helpful for understanding the Cloud Computing environment, architecture, how it works and how the inter-object of clouds (Datacenter, Broker, Host, etc...) Interactive together. In order to understand and analyze the provisioning (scheduling and allocation) algorithms (policies) behaving, designing and interaction, which has been very useful to design our VM allocation policy (Agent-based VM allocation policy) the main subject of this thesis, it is illustrated in the next chapter (Chapter 4).

Figure 3.1 shows the breakdown of the CloudSim toolkit classes and how the inter-objects of the cloud computing environment are modeled and implemented. Thus, the understanding of how the CloudSim toolkit is modeling and implementing the cloud computing, it makes the designing, implementing and testing new provisioning policies, such as the VM allocation (placement) algorithms (policies) more easily and applicable undertaking.

According to Figure 3.1 and the CloudSim toolkit designing classes (source code) [15, 35-37], The CloudSim toolkit environment architecture (Cloud Computing) has the following characteristics:

- Any Cloud Computing system (environment) has two sides; Broker side (who works on behalf of the users) and cloud side (datacenters or cloud providers).
- Broker side (DatacenterBroker class [38])
    - Works on behalf of the end users, companies and organization.
    - Receives a list of requested VM's from the end users.
    - Receives a list of Cloudlets (applications) to run them by the VM's.
    - It's in charge of binding the cloudlets with the VM's. So that anyone who is interested in designing and modeling a new algorithm or policy of distributed the Cloudlets over the VM's, he must override some methods inside the DatacenterBroker class based on his proposed algorithm (policy).
    - It's in charge to request from the Cloud; how many pooling resources (datacenter's) it has. Which is very vital for requesting, creating and distributing of the VM's and Cloudlets.
- Cloud Computing side (Datacenter class [39])

- The Datacenter has one list of hosts (computing nodes or servers).
- Each Datacenter has just one VM allocation policy, the CloudSim toolkit offers a default VM allocation policy, it's called VM Allocation Policy Simple [40]. Thus, any work interests of the VM allocation policies such as this thesis, it has to create own policy either by overriding the default CloudSim toolkit policy or creating a new one from the scratch.
- It has at least one list of created VM's and list of Cloudlets.
- Each Datacenter has own characteristics (DatacenterCharacteristics class [41]).

- The Host (Host class [42])
  - Even it is called the server or computing node.
  - It's in charge of creating, hosting and scheduling the created VM's. Hence it provisions all the necessary resources and algorithms (policies) for all previous.
  - It has at least a list of created VM's and list of Cloudlets.
  - Each host has own resources such as PE (Processing Element or Processor), RAM, Bandwidth, Storage and etc...
  - Every host uses some provisioning algorithms (Provisioner) to allocate and provision the resources to the VM, a consequence the VM allocation policy depends on these algorithms to perform its work, like:
    - RAM (Random Access Memory) Provisioner.
    - Bandwidth Provisioner.
    - MIPS (Millions of instructions per second) Provisioner.

Note that, the CloudSim toolkit provides a simple (default) provisioning algorithms (provisioners) for the MIPS, RAM and Bandwidth.

- Each host has at least one VM scheduler to handle the scheduling of the host resources, especially the computing cores (PE) between the VM's, moreover this scheduler it has own impact on the VM allocation process.
- The CloudSim toolkit provides two default VM schedulers (Time-Shared [17], Space-Shared [18]), which will illustrate later (Section 3.6).

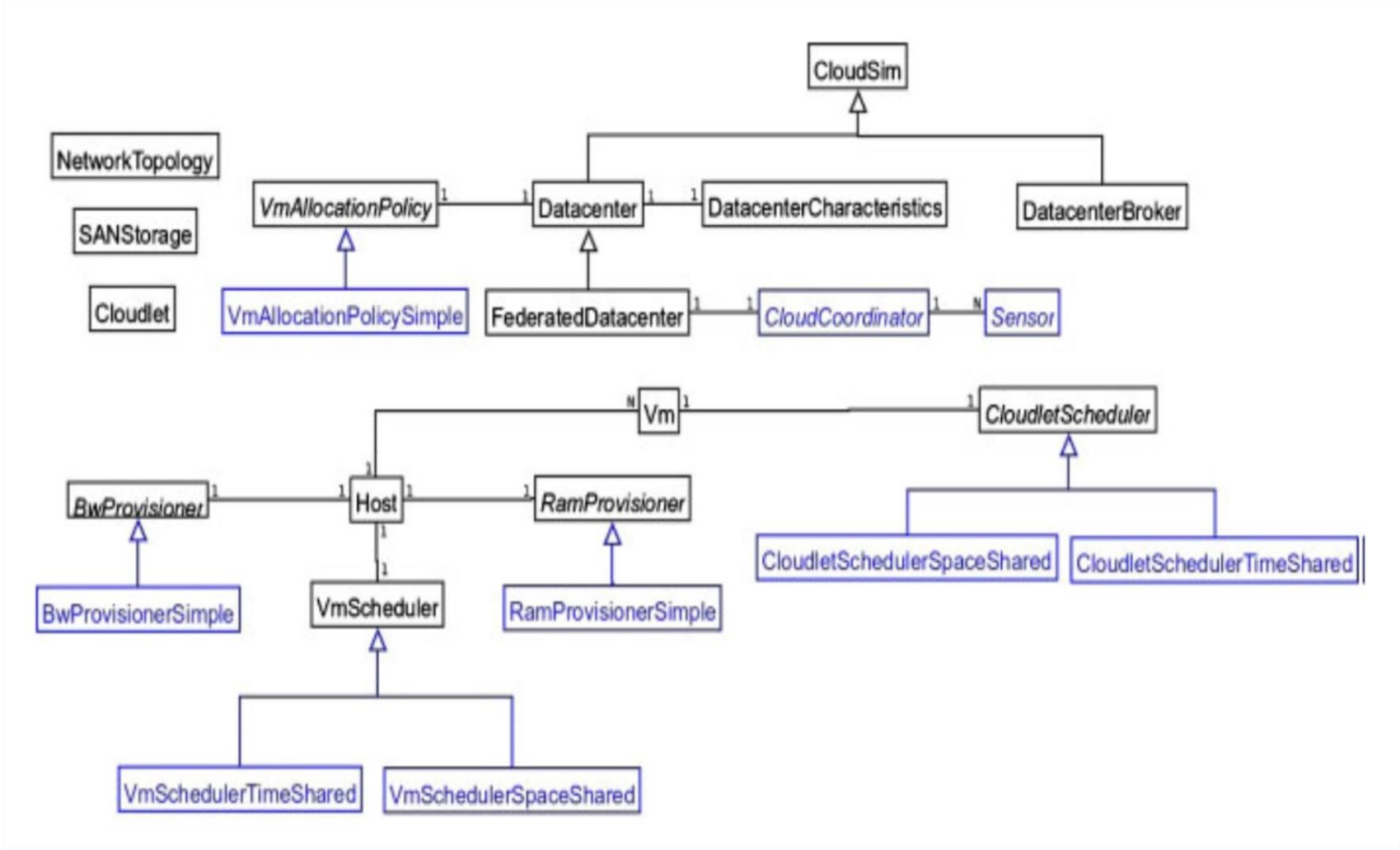

Figure 3. 1: CloudSim Toolkit Classes Design, taken from [15].

- The Virtual Machine (VM class [43]) is in charge of cloudlets (an application) execution, where it has a list of them and at least one Cloudlet Scheduler (Time-shared [44], Space-Shared [45]). In addition, each VM has a particular specification (RAM, PE's, bandwidth, MIPS etc...) must be fulfilled by the host before crating it.

In this section we tried to present an overview of the Cloud Computing environment, according to the CloudSim toolkit architecture, also a spotlight on the main parts those have main roles during the VM lifecycle (next section) more precise within the allocation (placement) processing.

3.4 VM Lifecycle (Allocation Dialog)

The VM as a part or element of Cloud Computing environment has a lifecycle, which is started from a Broker request for creating the VM inside the Cloud-Datacenter until the VM destroyed according again to the Broker request for that. In this context, a lot of Cloud Computing elements are involved during this progress, for instance, Broker, Datacenter, Host, etc.., also many algorithms (Provisioners), policies and schedulers are involved too.

In this thesis context, we focus on the VM allocation progressing, which is part of the VM lifecycle. The VM lifecycle depends on a live dialog between the Broker-side and Cloud-side which is illustrated by the sequence diagram in figure 3.2. The VM allocation dialog between Broker-side and Cloud-side (VM lifecycle) steps (figure 3.2):

- 1.1. First of all, the broker receives the list of VM's and list of cloudlets; then it sends a request to create VM's to the Datacenter; by sending the VM's features through a VM's object in the CloudSim toolkit (VM class [43]).
- 1.2. Then, the Datacenter calls the VM allocator (VM allocation policy [40]) to find and identify the proper host based on the VM specifications, host resources and the VM allocator technique.
- 1.3. After the VM allocator specified and identified the proper host, The Datacenter asks through the VM allocator from the host to create the current VM.
- 1.4. Now, the host attempts to create the current VM then send back an Ack about the creation attempt to the VM allocator. According to the Ack either repeat the step number 1.3 if it is failing or go to step number 1.5 if it is succeeding.

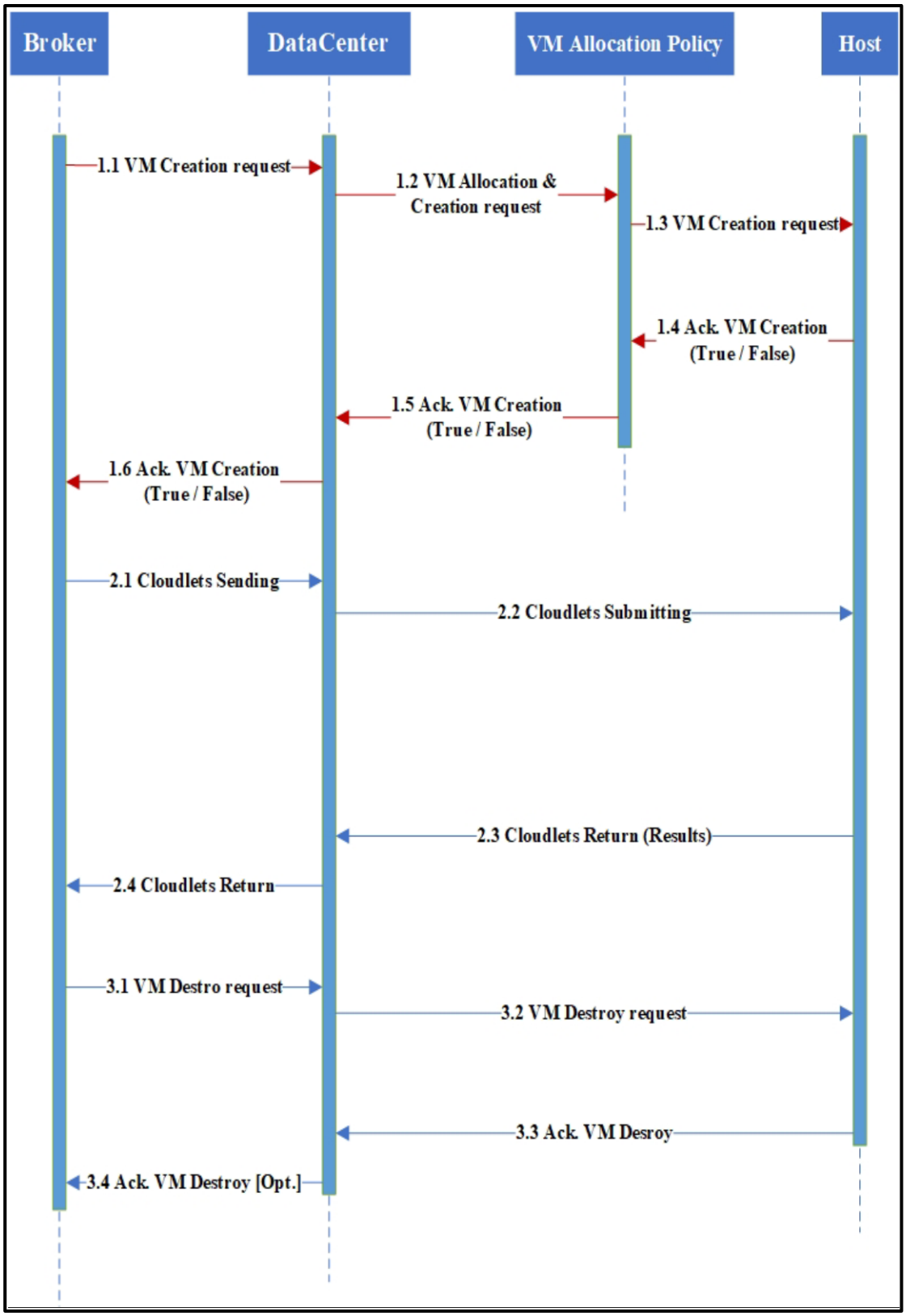

Figure 3. 2: The Sequence Diagram of VM Lifecycle.

- 1.5. The VM allocator sends the successful Ack back to the Datacenter. But in some cases it sends the failed Ack; after checking all hosts inside the datacenter without finding any suitable host for the current VM.
- 1.6. The Datacenter sends the creation attempt Ack back to the Broker.
- 2.1. The Broker, after receiving the last Ack from the Datacenter, starts sending the Cloudlets (data, application) to the Datacenter to run (execute) them on the VM's.
- 2.2. The Datacenter submits the cloudlets to the VM's to execute them according to the Broker distribution.
- 2.3. When the host completes the cloudlet's execution through the VM, it sends the results back to the Datacenter.
- 2.4. The Datacenter just passes the results to the Broker.
- 3.1. If the Broker has no more using to the VM; the Broker requests from the Datacenter to destroy the VM.
- 3.2. The Datacenter passes the destroying request to the host to destroy the VM.
- 3.3. After the VM is destroyed (terminated), the host sends back an Ack to the Datacenter.
- 3.4. Due to the Broker's request in step 11; the Datacenter sends back an Ack to the Broker or not.

Note that, any VM allocation policy takes place at point 3 of the allocation dialog (VM lifecycle) for finding the best choice from the hosts to the requested VM, based on the VM allocation policy techniques and criteria for allocation.

In the context of this thesis, proposing a new Agent-based VM allocation policy [19] (Chapter 4). The efficiency of the VM allocation policies can be measured by the ***Allocation Time***, the amount of ***Occupied Resources*** through the VM's, the ***VM's Distribution***, ***Datacenter Awareness*** etc.... Based on that the comparison between the VM allocation (placement) policies become easier and more standardized.

The ***Allocation Time***: refers to the amount of time to complete the allocation and placement of the VM to proper host, including the host identifying and VM creation i.e., the time duration starts from the moment that the Broker sends a request of VMs creation (step 1.1, figure 3.2), until the moment that the Broker starts sending of cloudlets (step 2.1, figure 3.2).

Note that, the Allocation Time covers some parts of the allocation dialog (the Red steps (1.1 – 1.6), figure 3.2), also it has own impact over the allocation dialog. Where if the Allocation Time is very long the allocation dialog will take a long time even. More specifically, the Allocation Time is considered as a part of the Response Time and the Turnaround Time, and as mentioned it has a serious impact on those performances and efficiency terms.

The ***Occupied Resources***: refer to all allocated resources such as the number of PEs, MIPS, RAM, BW and Storage, over datacenter's hosts, for creating the requested VMs through the hosts in the datacenter.

Therefore, the efficient behavior among the Allocation Time, Occupied Resources etc. gives a certain VM allocation (placement) policy the advantage over the other policies.

After the VM lifecycle (allocation dialog) is presented during this section according to the CloudSim toolkit point of view. Moreover, some efficiency and performance of the VM allocation policies are presented and defined; the next section presents some terms about the VM creation progress and the impact of the VM allocation policy on it.

3.5 VM Creation and Allocation Issues

This section presents some important issues about the VM specifications, which have a serious impact, firstly on the resources (host) allocation, then VM creation in these resources. Add to that, present some issues about how the host creates the VM and provision the resources to it.

According to the VM class [43], each individual VM has special specifications which formalize the features, requirements, size, price etc. of the VM. Those specifications are passed or specified as parameters to the VM class constructor method, which means the constructor method during the construction process for the VM object; defines the specifications as values to these parameters.

According to the table 3.1, the VM parameters can be classified into three categories. Firstly, parameters have no impact on the allocation and creation progress, such as VM Id and User Id. Secondly, parameters important to the allocation and creation progress, but do not need to the provisioning algorithms or scheduler during the creation (placement) progress like, VMM, CloudletScheduler and sometimes the size parameter. Thirdly, parameters very

important during the allocation and creation (placement) progress of VM, moreover, they need to special provisioning algorithms or scheduler to complete the allocation and the creation process as the rest parameters (MIPS, NumberOfPes, RAM, BW) [15, 37].

The VM constructor [43] takes the following parameters, Table 3.1:

Table 3. 1. VM class parameters.

Parameters	***Meaning***	***Function***
id	The VM Id	Each VM has unique Id, differentiate the VM
userId	The user Id	Each VM belongs or serves just one user
mips	Million Instructions Per Second	Declare the speed of the processor(s)
numberOfPes	Number Of Processing Elements (PEs)	Identify the number of processing cores of the host
ram	Random Access Memory (RAM)	Identify the amount of RAM needed
bw	System bus Bandwidth (BW)	Define the amount of the system bus Bandwidth needed for the VM
size	The size of the VM	Specify the storage capacity (size) of the VM
vmm	Virtual Machine (VM) Monitor	Identify the Virtual layer type or toolkit, the VMM type.
CloudletScheduler	The Cloudlet Scheduler	Identify which Cloudlet Scheduler will use inside the VM for cloudlets

In the context of the VM allocation (placement) Policies, each VM feature (parameter) has an impact or very important during the allocation; it is considered as a VM specification. While each VM specification is considered as one *Dimension* to the VM allocation policy

through the searching and identifying among the datacenter hosts. The VM allocation policies are available with different levels of the verifying as One-Dimensional, Two-Dimensional etc.., Hence, more dimensions in the VM allocation policy means more accurate allocation decision, good performance and short allocation time.

The allocation progress depends on the number of dimensions and the selection criteria of the VM allocation policy. First, the host verifying and checking progress based on the number of dimensions in the policy; which means how many specifications (resources) inside the host will verify corresponding to the VM needs such as RAM, BW, etc... Second, the VM allocation policy selection criteria, for instance, the First-Fit, Worst-Fit (Max-Rest) etc.... Consequently, the VM allocation policy selects the proper host according to its procedures and techniques.

Table 3. 2. The Host resources and provisioning algorithms, [42].

Resource	***Provisioning Algorithm or Scheduler***	***Description***
PE	VM scheduler [46] + PE Provisioner [47]	Processors are allocated through the VM scheduler and PE provisioner.
MIPS	VM scheduler [46] + PE Provisioner [47]	Because the MIPS is belonged to the PE, each PE has a specific MIPS for itself.
RAM	RAM Provisioner [48]	Allocate the requested RAM capacity to the VM
BW	System bus BW Provisioner [49]	Provision and allocate the needed system bus Bandwidth to the VM
Storage	******	The host allocates the Storage Capacity directly.

After that, the VM allocation policy requests from the selected host to start the creating process of the VM. In the CloudSim toolkit, the VM allocation policy passes the object of

the VM to the selected host to create it; so the host will try to provision all the VM object needs from the available resources through the provisioning algorithms and scheduler. Table 3.2 shows host resources and provisioning algorithms.

As mentioned, the selected host will try to provision the available resources during the creation operation through the provisioning algorithms (provisioners). CloudSim toolkit offers default provisioners for specifications in table 3.2 such as: (i) PeProvisionerSimple Class [50] for the PE, (ii) RamProvisionerSimple Class [51] for the RAM and (iii) BwProvisionerSimple Class [52] for network BW. In addition, the VM scheduler is used to schedule and allocate the PE's and MIPS resources between the VMs inside the host, where the CloudSim toolkit offers two VM scheduler to manage the resources among the VMs (Time-Shared [17], Space-Shared [18]), those schedulers will explain in the next section.

The VM creation progress in the host (based on CloudSim toolkit) takes the following sequence [42]:

1. Allocate storage capacity equal to the VM size, if there is not enough storage space terminate the creation progress (exit).
2. Then, allocate the needed RAM to the VM through using the RAM provisioner, if the RAM is not fulfilling the VM need do: deallocate the storage capacity then finish the creation progress (exit).
3. After that, allocate the system bus BW for the VM by the BW provisioner; if there is not enough system bus Bandwidth do: deallocate the allocated (provisioned) RAM and storage then cancel the creation progress; else continue (exit).
4. Finally, allocating the PEs and MIPS simultaneously through the VM scheduler, which depends on the PE provisioner to perform that. Again, if there are not enough resources to complete the allocation of the requested PEs and MIPS the host does: deallocate the allocated BW, RAM and storage, then terminate the creation progress (exit).

Aforementioned, the succeeded VM allocation policy, which puts in its considerations: (i) the VM specifications (Dimensions), (ii) how to use the provisioning algorithms in an efficient way and (iii) how the VM creation progress is made in the host.

Finally, this section presented some important issues about the allocation and creation progress based on the CloudSim toolkit (cloud computing environment). The next section will present a brief idea about the VM Schedulers and their impact on the VM allocation policies work (allocation progress).

3.6 The VM Schedulers

The Virtualization layer it is an extra layer in the Cloud Computing comparing with the conventional computing platforms for the application services. Thus, it is one of the most advantages against the Grid Computing (one of conventional computing nodes); which performs a management, execution and hosting for the application services [4, 15, 16]. The Cloud Computing offers through the extra layer (VMM) a computational abstraction resources provisioning to the application services, besides a computational abstraction the VMM layer guarantees the isolation between VMs during the instantiating process in a host [4, 15, 16].

Practically, each VM in the host is fully isolated over the physical and secondary memory space, but the processing cores and a system bus (Bandwidth) are shared with the other VMs in most cases. Therefore, the available amount of processing and a system bus (hardware resources) to each VM is forced by the total available amount of processing power and system bandwidth inside the host during the VM provisioning process. Thereby, every processing power demand through the VM creation progress more than the maximum processing power of the host is denied, and the creation progress will cancel [4, 15, 16].

Aforementioned, the CloudSim toolkit offers two levels for provisioning and scheduling the processing power between VMs: (i) the host level ; in this level a proportion of the overall process power of each processing core is specified and assigned to each VM, (ii) the VM level [53]; in this level the overall process power of each VM is distributed among the application services (task units) that are already hosted within its execution engine. Not that, the CloudSim toolkit builds two schedulers (provision policies) for each level (Time-Shared, Space-Shared) [4, 15, 16].

This thesis concerns on the VM allocation policies and any other algorithms involved or effect on the allocation progress. So, in this context, the scheduling in the first level (host-

level) is very significantly more than second level (VM-level), because it is between the VMs and it has an impact on the allocation policies work.

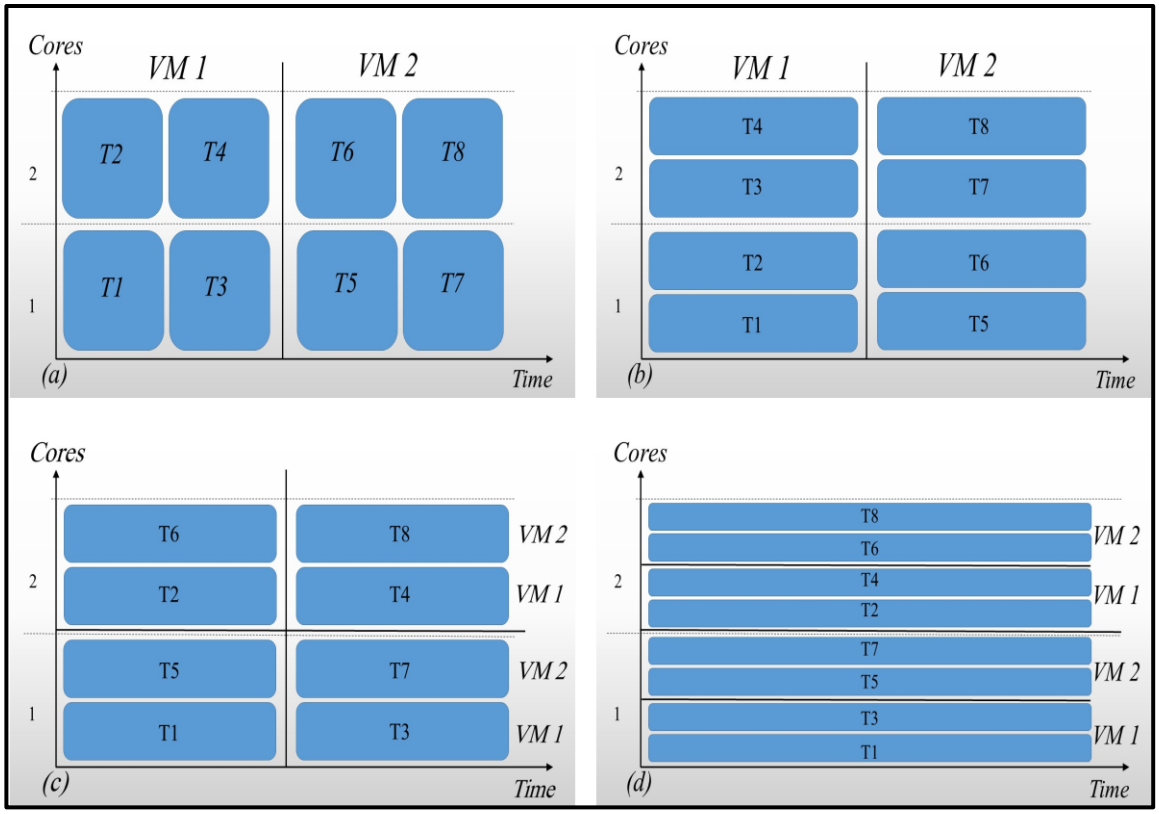

Figure 3. 3: The impacts of the schedulers (provisioning policies) on task unit execution. Taken from [15].

Figure 3.3 gives an example of the two types of the scheduling (provisioning policies) on the two levels. The example about, one host with two PEs (processing cores) which is hosting two VMs to execute 8 tasks (application units) as: t1, t2,…, t8. The tasks are distributed on VMs equivalently; t1-t4 to the first VM and t5-t8 to the second VM.

Moreover, figure 3.3 shows the four possible cases for the scheduling (resources provisioning) between the VMs and cloudlets (tasks, application units). Cases as follows:

a) Using the Space-Shared scheduler for VMs [18] and cloudlets units [45] (same one to the two-levels).
b) Using the Space-Shared scheduler for VMs (first-level) [18], and Time-Shared scheduler to Cloudlets units (second-level) [44].
c) Using the Time-Shared scheduler to VMs (first-level) [17], and Space-Shared scheduler to the Cloudlets units (second-level) [45].

d) Using the Time-Shared scheduler to VMs [17] and Cloudlets units [44] (same one to the two-level).

According to the figure 3.3 at the first level (context of this thesis); the time-shared scheduler gives the opportunity to more than one VM sharing the same host processing cores (computing resources) in a dynamic fashion. Consequently, the scheduler distributes processing power amongst VMs based on time-shared slices. The latter means each VM is getting a time portion to use the processing power of host cores, like (c) and (d) in figure 3.3.

On the other hand, the space-shared scheduler is contradictory with the time-shared scheduler as it assigns host processing cores (computing resources) to a particular VM in a static fashion, i.e. the scheduler prohibits any sharing of these cores with other VMs. The assigned processing host cores serve just one VM during its life cycle, like (a) and (b) in figure 3.3. Thus the number of created VMs by using the space-shared scheduler is less than the time-shared scheduler under the same conditions; that will be very obvious through the numerical experiments results in Chapters 5, 6 and 7.

Finally, the VM schedulers have own impact on the VM allocation policies such as the number of created VMs as mentioned before and on the VM allocation policies behaving manner through the allocation progress as comes in following chapters.

3.7 The Default VM Allocation Policy

The CloudSim toolkit offers a default VM allocation policy, which is called *VM Allocation Policy Simple* [40], to help the users or researchers of the CloudSim toolkit to perform the allocation progress through their works or researches and to develop a new VM allocation policies (provisioning algorithms). Consequently, this VM allocation policy it is very simple and based on a simple algorithm, which depends on just one dimension (one VM specification) to identify and allocate the proper host according to its techniques.

The algorithmic steps of the *VM Allocation Policy Simple* [40], are as follows in Algorithm 3.1.

Algorithm 3. 1. The VM Allocation Policy Simple.

Algorithm 3.1 *VM allocation policy simple*

Variables: *VM_Table Map*, *Used_PEs* List, *Free_PEs* List.
Input: ***VM*** Object
Output: *Boolean* ***Result***

```
1   Result = false / Tries = 0;
2   If (VM is not already created)
3       Do:
4           For Each (host in Host_List)
5               Find the host with more free Pes;
6               Return the index;
7           End For
8           Get the Host. Id = index;
9           Result = Create_VM (Host);
10          Tries ++;
11          If (Result)
12              Update:
13                  VM_Table (VM)
14                  Used_PEs (required_Pes)
15                  Free_PEs (required_Pes);
16              Result = True;
17          End If
18          Else: set the Host in minimum value.
19      While (! Result && Tries < Free_PEs. Size)
20  End If
21  Return Result;
```

Based on Algorithm 3.1; the default policy is considered as one-dimensional VM allocation policy, where it depends just on one VM specification during the allocation progress, which is the number of PEs (c.f. Table 3.1). Also, all VM allocation policies try to create the VM instance directly after finding the proper host, so either the creation progress is successfully done, then the policy returns a *True* result or failed then the policy returns *a False* result.

Finally, this default policy will compare with the Agent-based VM allocation policy (the core of this thesis) under the same conditions through different scenarios in Chapter 5.

3.8 Summary

This chapter conducted a review of the CloudSim toolkit, which highlighted the advantages of using the simulation tools, in general, and the CloudSim toolkit, in particular. In this context, the Cloud Computing environment through the CloudSim toolkit structure with classes' breakdown was presented. Moreover, it explained the VM allocation progress through the VM lifecycle (allocation dialog), the most significant issues of allocation and creation processes and the VM scheduler types and their impacts. Finally, the default VM allocation policy was presented.

The following chapter introduces the proposed Agent-based VM allocation policy.

Chapter 4 A New Agent-based VM Allocation Policy

4.1 Introduction

In the Cloud Computing, computational resources are composed of multi-thousands of physical machines (PMs) inside a datacenter such as the Microsoft, Amazon and Google datacenters; to host the VMs that process the end user's and client's requests [5, 6]. Therefore, the VM allocation (provisioning) policy is very important and it has a significant role to find and allocate or "place" a VM in an appropriate Host (PM: physical machine) according to the VM requirements and specifications. So that, the VM allocation (provisioning) process [15, 31] can be defined as: it's the process of identifying the host (PM) that corresponds the critical properties (storage, memory), configurations and requirements of particular VM; then creating an instance of it in that host. Hence, the VM allocation policies can be classified into two main categories: the first type provisions and places VMs on the proper hosts according to requests received from brokers and end users, and the second type optimizes the current allocation of VMs [7, 8].

Therefore, designing appropriate methodologies and implementation techniques that can allow and adjust the dynamic behaviors of cloud computing systems; it is a hot research field. Besides, the use of multi-agent technology in large-scale data centers in order to search, allocate, provision and update the applications, resources (i.e. VMs) and massive volumes of data. That will provide intelligent allocation policies, provisioning algorithms, monitoring services, data access services and energy-efficient use of Cloud computing infrastructures [14].

This thesis presents a new ***Agent-based VM allocation Policy***, which employs the multi-agent system technology through the VM allocation process; to achieve efficient allocation results such as an Allocation Time, Occupied Resources and VM's distribution (c.f. Section 3.4). Moreover, *Agent-based VM allocation Policy* belongs to the first category of VM allocation policies, where in the context of this thesis it just interests in the VM allocation (provisioning) and placement.

In addition, most of current algorithms for placement 'allocation' policies belong to the second category of allocation 'placement' policies [6, 7], which focuses on the optimization of placement 'allocation' during the VM's lifecycle e.g. policies in [9-12] for a particular purpose,

such as an energy efficient and SLA. Also, the majority of the current state of the art policies is depending on traditional (conventional) techniques to perform the allocation process.

In the rest of this chapter introduces the communication and coordination approaches of the multi-agent system, the proposed multi-agent system for Datacenter, the testing criteria and the algorithm design and implementation of Agent-based VM allocation policy.

4.2 Communication and Coordination Approaches of the Multi-Agent System

This section introduces brief overview about some communication and coordination approaches for the Multi-Agent system. Whereas, the Agent does not work alone in the environment, but it works as a member of a group (multi-agent system). Therefore, communication is an important component of a multi-agent system; because the agent needs to contact and communicate with the users/owners, the resources of the system and other agents. Further, Agent needs a special communication language to do that, which is known as an agent communication language. This language is defined based on the speech act theory (*Searle, 1969*)/[54] and offers isolation between the communicative actions and content language [55].

Additional, the first widespread agent communication language was KQML (*Mayfield et al., 1996*)/[56, 57], which was developed as a part of the US government's project in the early 1990s. Nowadays, the most commonly used agent communication language is FIPA's (the Foundation for Intelligence, Physical Agent), ACL (Agent Communication Language). Whereas, the main features of FIPA - ACL is the ability to use many different content languages, and the ability to use a predefined interaction protocol for managing a conversation between agents [55].

The second attribute for teamwork in a multi-agent system is the coordination, to ensure that the whole system will work in a coherent and consistent manner by involving all the agents together. There are many reasons for coordination needing in a multi-agent system [55, 58]:

1. To prevent conflicts between agent's actions, which are based on agent's goals.
2. To avoid any congestion in the system, in case there are any dependencies between the agent's goals.

3. To ensure good load balancing in the system, especially if there is a difference in the capability and knowledge between agents.
4. Coordination increases the efficiency of the system for achieving the agent's goals.

Agents can handle coordination through a number of approaches, some of which are described below [55, 58]:

- ***Organizational Structuring***: this approach presents a scope for the interaction and activity via the definition of roles, communication paths and authority relationships *(Durfee, 1999)*/[59]. This approach is a simple coordination approach to prevent conflicts and ensure system coherency. In addition, it is based on a master/slave or a client/server model (central controls). However, in real applications, this approach is difficult to create and is contrary to the nature of the multi-agent system nature (decentralized or autonomous).
- ***Contracting***: The contract net protocol (*Smith and Davis, 1981*)/[60] is one of most important coordination techniques used to allocate tasks and resources, among agents and in determining the organizational structure. This is a decentralized approach in which an agent can either be a manager or a contractor. However, the main principle of this approach, is that if any agent fails to solve the allocated problem using local resources/expertise, the problem can be decomposed to a sub-problem and passed to the interested agent with sufficient resources/expertise. The contracting mechanism to solve the work assignment problem c.f. Figure 4.1: (1) the manager agent announces the contract/task; (2) the contracting agents' response to the announcement by sending back their bids; (3) a manager agent evaluates the submitted bids, then, according to the most appropriate bids evaluation, it grants a sub-problem contract to contractor(s).
- ***Multi-agent Planning Problem***: in this approach, agent coordination is achieved by building a plan, which explains all future actions and interactions, that are required to achieve all agents' goals, planning and re-planning. There are two types of multi-agent planning: (a) *Centralized multi-agent planning*: In this sub-model, there is usually a coordinator (coordinating agent) in the system, which receives all partial or local plans from agents. After, receiving all the local plans, the

coordinator will verify and validate all plans to check for inconsistencies and conflicting interactions. Hence, it decides where/which the appropriate modifications will be applied to solve any conflicts (*Georgeff, 1983*)/[61]. (b) *Distributed multi-agent planning:* There is no coordinator in this model, whereas the responsibility of forming/composing the global plan for the whole system belongs to all agents. Therefore, the agents use interleaves communications to build and update their individual plans and their suggestions (models) for other agents, until all inconsistencies and conflicts are eliminated (*Georgeff, 1983*)/[61].

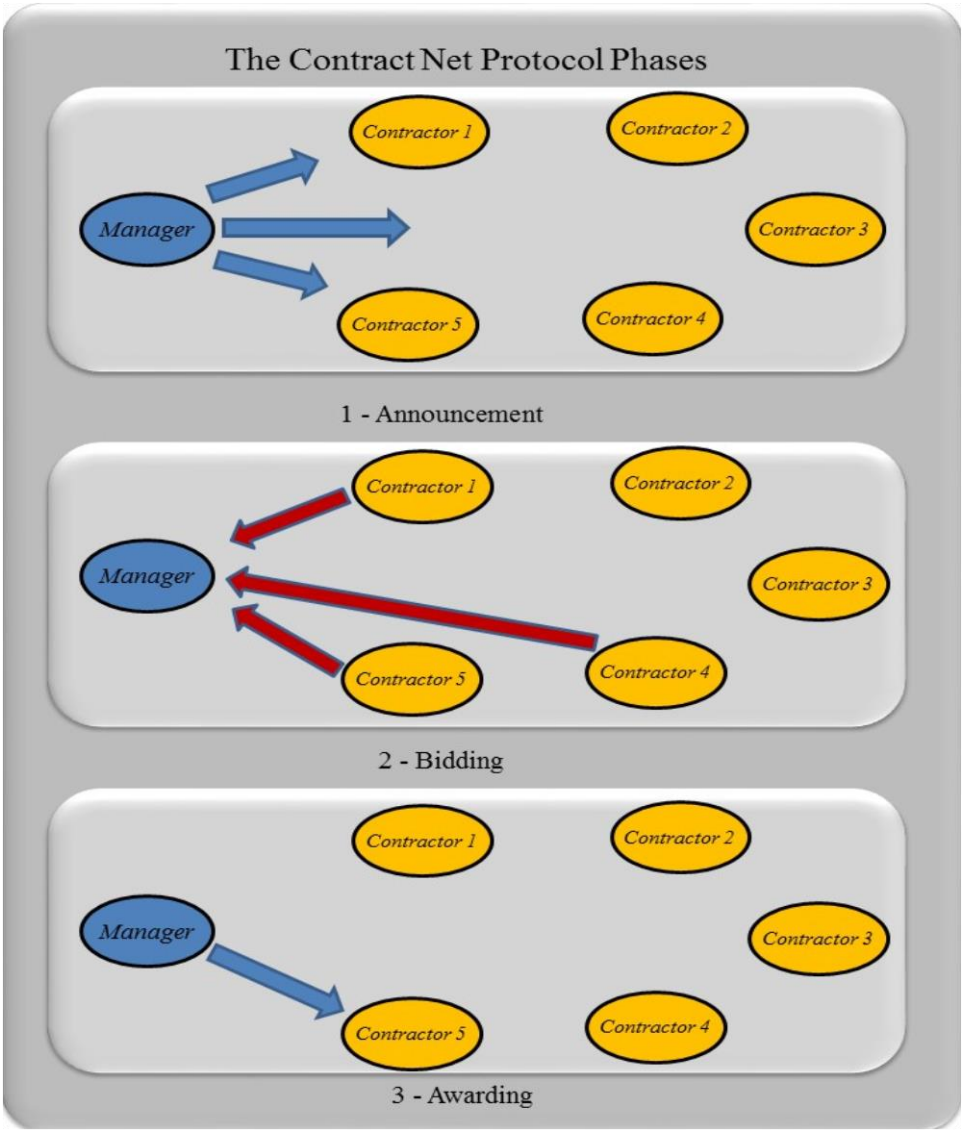

Figure 4. 1: The Phases of The Contract Net Protocol, Taken from [55].

- ***Partial Global Planning***: This is not a new approach, but an integration of strength for the three previous approaches: Organizational structure, Contracting and Multi-agent planning by uniting them in one approach (*Durfee and Victor, 1987*)/ [62-65].

- ***Negotiation***: This technique is supported by most agents. Where, the communication operation between a group of agents for reaching a mutually accepted agreement on some issues or matters is called, a *Negotiation* (*Bussman and Mullar, 1992*)/[66]. Additional, the Negotiation is based on the nature of the agents involved and goals that should be achieved. However, there are two types of negotiation :(a) Competitive negotiation; is used if there is no common/global goal among all the agents, but there are independent individual's goals. In such cases, there is no requirement to cooperate or share information; which means a relationship between the agents, is competitive in nature, (b) Cooperative negotiation, is contrary to the competitive negotiation; because, there is a global goal that should be achieved by all agents. Therefore, cooperation and sharing of information are necessary. More specifically, in such cases a multi-agent system is usually designed as, a central architecture to achieve a single common goal.

Finally, next section presents the proposed Multi-Agent system for the cloud datacenter including: the conceptual structure components and the coordination approach for the entire agents.

4.3 The Proposed Multi-Agent System for Virtualized Datacenter

Using of a multi-agent system is considered as a core of the Agent-based VM allocation policy, where the allocation policy uses the multi-agent system through the allocation process to search and identify the proper host to the requested VM. Every single multi-agent system has at least special structure and special communication and coordination system among the member agents of the system.

4.3.1 The Structure of the Proposed Multi-Agent System

In the context of this thesis, the multi-agent system is proposed to use in the VM allocation policy to search and find a suitable host to the current requested VM. Hence, the structure of a multi-agent system in our policy is adaptable and adjustable to datacenter architecture design (network topology); thus the system structure can be small and simple or huge and complex based on the size, resources and design model of the cloud datacenter.

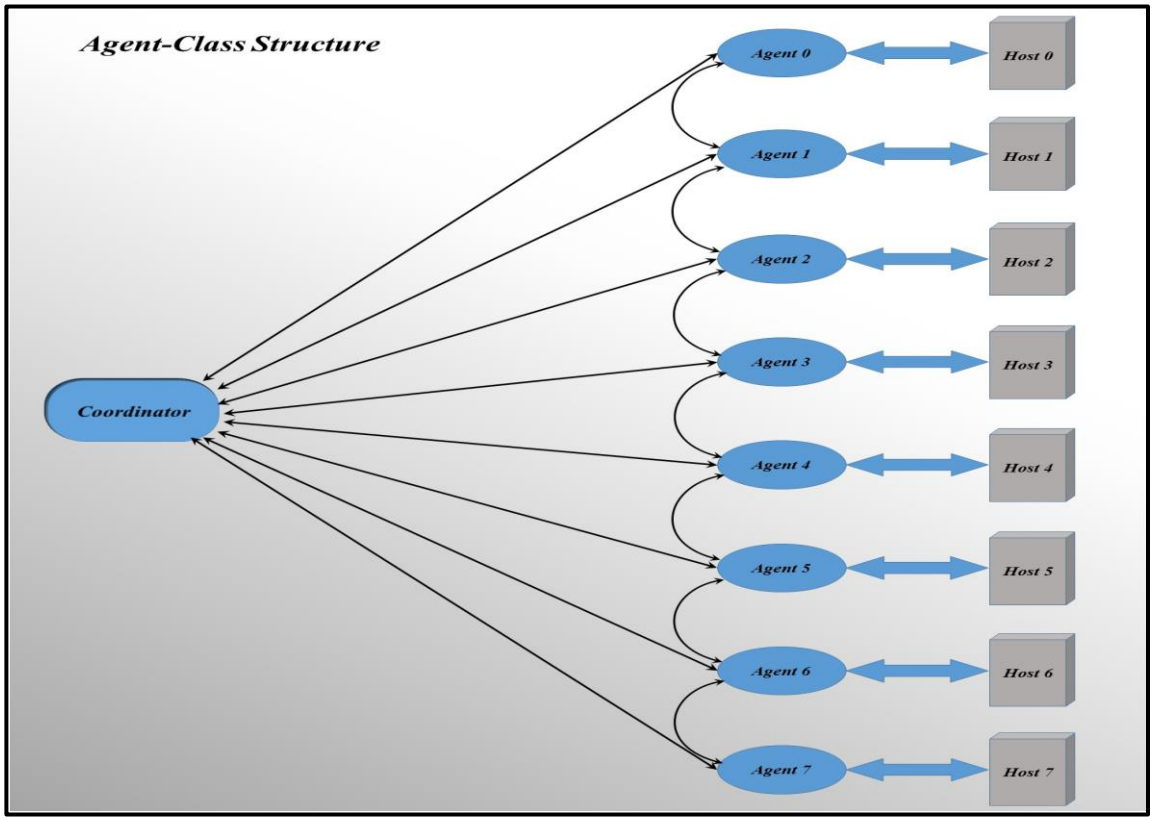

Figure 4. 2: Agent-Class Structure.

There are two aspects to describe the multi-agent system structure in our VM allocation policy: *Agent-Class* c.f. figure 4.2 and system's layers c.f. figure 4.3. Consequently, the conceptual structure of a multi-agent system consists of small structural units called classes or groups or flocks, which are distributed over multi-layer hierarchy through the system. The system has at least one *Agent-Class* with two layers; due to each *Agent-Class* has two layers, or a system has more than one class according to the datacenter size. In addition, an agent-class

has two agent types: (i) coordinator-agent (layer 0) and (ii) member-agent (layer 1) c.f. figures 4.2 and 4.3.

The multi-agent system architecture for this policy is based on a *Partial Global Planning* [62-65] approach combined with the *Negotiation* [66] approach (c.f. Section 4.2) as described below:

1. Each Host is connected by an Interleaf-Agent c.f. figure 4.2, at last layer (level) a member-agent called interleaf-agent; which can test and verify the host available resources for the requested VM.
2. The agents are collected together to form an Agent-Class or a flock (group) of agents, c.f. figure 4.2.

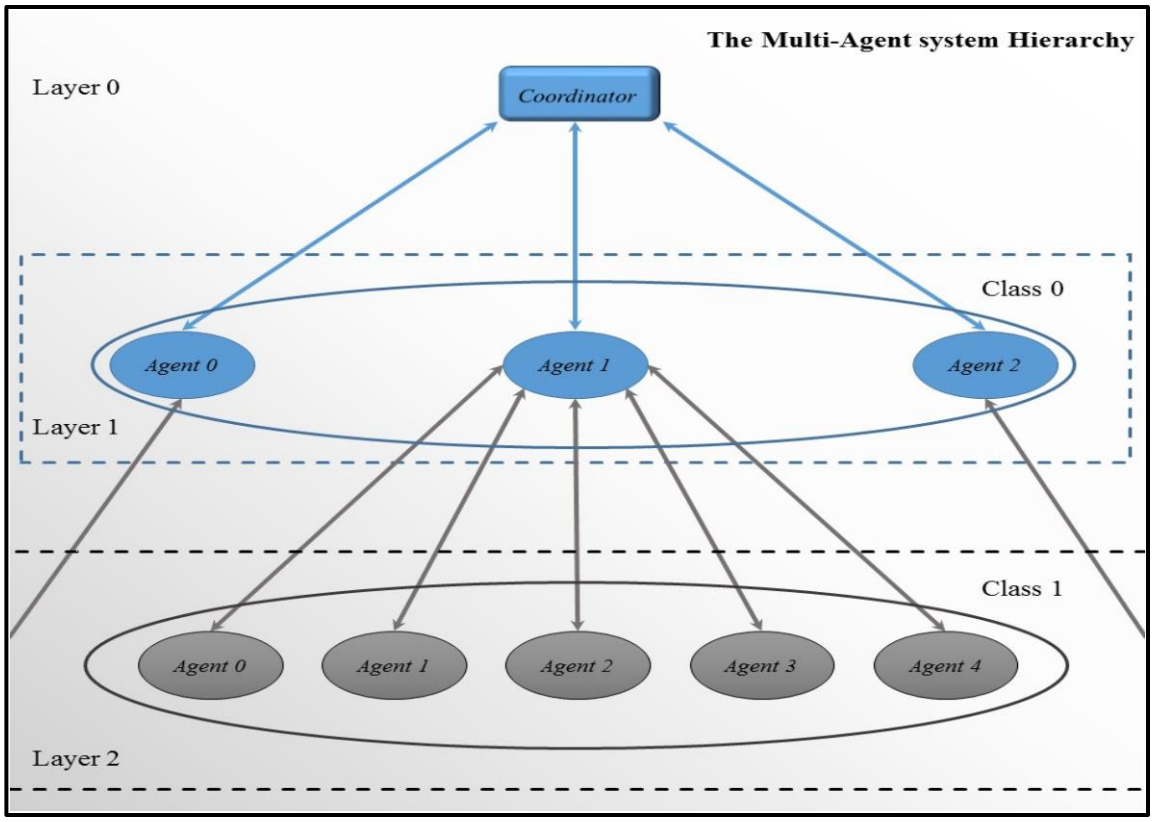

Figure 4. 3: The Multi-Agent System Hierarchy.

3. Each class has a special agent, which works as a coordinator of the class/flock (c.f. figure 4.2), during the execution a global task over the class such as the allocation process; to optimize the performing of member-agents and prevent any conflict between them.

4. The system in general consists of a hierarchy classes c.f. figure 4.3, where each class is distributed over two layers: one for the coordinator and another one for member-agents.

5. The coordinator agent of any class is a member of a higher layer class. This means that the coordinator agent of class in layer (n) is a member-agent of another class in layer (n-1) under the supervision of another coordinator (a higher class coordinator), figure 4.3.

6. The multi-agent system overall structure is based on the datacenter architecture design model, which means the number of layers, agent classes, coordinators and agents depend on the data center architecture design, resources and size of it.

7. The *Contract Net Protocol* is used as a communication and coordination system through performing a global task, such as a VM allocation process in the context of this thesis; which means the system uses the contracting to find and identify the best choice from datacenter's hosts for the requested VM.

8. In general, the contracting or contract net protocol might be occurred either from coordinator to a member-agents or between the member-agents directly.

9. Sharing of information and results are possible between the member-agents under the coordinator supervision. Cooperative negotiation is required for solving problems and planning.

10. There is the possibility of competitive relations (competitive negotiation) between Agent-Classes (group vs. group) based on the situation.

Note that, Points 1 to 6 illustrate the system structure and how the agents are integrated with the system architecture. Points (7, 8, 9 and 10) describe the relationships and how agents work together as a system to get a global solution.

After the introducing of the proposed multi-agent system to our VM allocation policy; the next sub-section will introduce a full explanation of the agent communication system (Contract Net Protocol) which is used in the policy during the VM allocation process.

4.3.2 The Agent Communication and Coordination System (Contract Net Protocol)

As mentioned in Section 4.2, a Contract Net Protocol [67] is one of the communication and coordination approaches of the multi-agent system; in order to handle and manage the agents among the system for performing the tasks and functions together without any conflictions. In addition, every multi-agent system must have a communication and coordination approach (system) for many reasons such as smoothly agent's corporations and management… etc. (c.f. Section 4.2).

The Contract Net Protocol [67], consists of three phases: (a) Announcement, (b) Bidding, and (c) Awarding. In the context of this thesis, a modified Contract Net Protocol is used between a datacenter coordinator-agent and interleaf-agents (hosts) during the VM allocation process; in order to identify and find a proper host for the request VM.

The VM allocation process based on a Contract Net Protocol (c.f. figure 4.1) in Agent-based VM allocation policy, can be described as follows:

- After, a datacenter coordinator, receiving the requested VM; the coordinator broadcast (***Announcement***) the VM's specifications (requirements) to the interleaf-agents (hosts).
- Each interleaf-agent (host-agent) verifies and tests the host available resources according to a special testing criterion (next section) to fulfill the VM's specifications. The testing criteria might be either the ability of the host to host the VM, the percentage of resources utilization, the history of the host… etc. or combination between two or more of previous factors through using a special weight-function.
- Each interleaf-agent (host-agent) sends the obtained result back to the coordinator (***Bidding***).
- The coordinator after receiving all results (Bids) from the interleaf-agents; it selects the best result (Bid) based on a special selection (choosing) criteria (next section).

The selection criteria correspond with the testing criteria in the interleaf-agents; which means they depend on same factor or combination of factors.

- Finally, (***Awarding***) the coordinator asks from the selected host to create the requested VM (hosting).

Note that the Contract Net Protocol *(Smith and Davis, 1981)* [67], it's not a new algorithm or technique in the context of MAS as it has been involved in several MAS-based applications and researching efforts such as in [68-73]. However, in the context of this work, the adoption of the modified Contract Net Protocol, which serves the VM allocation processing through the cloud computing data center, it is considered as an innovative idea.

4.4 The Algorithm testing and selection criteria

In terms of the Contract Net Protocol, the Agent-based VM Allocation Policy has a special testing and selection criteria; which consists of two phases (levels). First one: on the agent-host level (Phase 1) and the second one: on the datacenter-coordinator level (Phase 2). On the one hand, Phase 1 defines the rules to each agent-host to decide either the host has the ability to create the VM or not. On the other hand, Phase 2 defines the rules of how the coordinator selects one of the passed hosts (from Phase 1) to create the requested VM.

Note that, in the context of this work two agent types are built; will describe in the next section of this chapter. First agent type is a datacenter coordinator agent, which is called Datacenter_Coordinator. Second agent type is an interleaf-agent (host-agent) which connects directly to the host and is called Host_Agent.

More specifically, the Phases 1 and 2 are operationally described below.

- Phase 1 (Testing-phase): Each Host_Agent checks his host if it has the ability to carry out the provisioning of the resources for the VM, then returns the result to the Datacenter_Coordinator as a Boolean value (True/False);
- Phase 2 (Selection-phase): The Datacenter_Coordinator puts all results (true/false) from a Host_Agent in a table, then in ascending order starts attempting to allocate and create the VM just for a host who has a true value.

The Agent-based VM Allocation Policy testing and selection criteria are considered as simple criteria. Where it takes into consideration just the host ability of provisioning resources to the

VM; due to the context of this work just aims to verify the concept of using a multi-agent system through the allocation process, and prove how much is successful. Moreover, the Agent-based VM Allocation Policy belongs to the first category of the VM allocation policies; which means it just interests in resource provisioning. In the future, the Agent-based VM Allocation Policy might be interested in the optimization (second category) and used more complex criteria.

It is assumed that under the auspices of the Agent-based VM allocation Policy, all the VM requirements (resources) are taken into consideration (full-dimensional). Therefore, the Agent-based policy does not try to allocate and create any VM in any host before checking that host if it can fulfil all the VM's specifications; this means less number of attempts is required, in comparison with other algorithms such as a default algorithm [40] of the CloudSim toolkit, for allocation and creation of VMs. This leads to an overall reduction of the allocation and turnaround times.

The next section will introduce the design and implementation of the Agent-based VM Allocation Policy and its algorithm through the CloudSim toolkit [36].

4.5 Design and Implementation of the Algorithm

This work applies the multi-agent system technology over the allocation progressing among the hosts (PMs) inside the cloud datacenter. This is achieved through the proposed new Agent-based VM allocation policy, which improves the performance of VM allocation policy by using some of the multi-agent system technology features and advantages.

Technically, a multi-agent system structure in datacenter through a CloudSim toolkit and practical experiments, it consists of two types of Make-Decision agents. The first type is a coordinator agent; called Datacenter_Coordinator and each datacenter has just one. The second type is an inter-leaf agent; called Host_Agent, one for each host.

In the context of this thesis, the system contains one Agent-Class with two layers because all datacenters through the practical experiments are in small scale, and all agents (coordinator and interleaf) are very simple in nature. Moreover, the system utilizes the contract net protocol [67, 73] (Figure 4.1) enabling the Datacenter_Coordinator to handle the group of Host_Agent

through the allocation progressing for finding the best suitable host to the requested VM (Section 4.2.2).

Based on the system structure and handling technique, the new Agent-based VM allocation algorithm aims to achieve a good awareness of datacenter resources (pooling of resources). This leads to efficient allocation decisions to prevent any failed allocation attempts (creation attempts) in the host (PM).

According to the efficient allocation decision, which prevents the failed allocation attempts, that aims and tries to improve the Allocation time of cloud datacenter during the allocation progressing and the allocation or occupied resources efficiently among the hosts (PMs) inside the datacenter over different scenarios (Chapters 5, 6 and 7).

The Design and Implementation of the Agent-based VM allocation policy take into consideration the CloudSim toolkit environment architecture. Thus, the Agent-based policy consists of three Java classes: (i) class of Vm_Allocation_Policy_Agent which is extended from the class of *Vm_Allocation_Policy* [74], (ii) class of Datacenter_Coordinator which is the responsible of a coordinator-agent object instantiation, and (iii) class of Host_Agent is in charge of interleaf-agent object instantiation.

According to the CloudSim toolkit environment, any new VM allocation policy might be taking place in it must be extended from the abstract class of VM allocation policies that called *Vm_Allocation_Policy* [74]. Consequently, Agent-based policy configuration through the CloudSim toolkit creates one object of Vm_Allocation_Policy_Agent and one object Datacenter_Coordinator for each datacenter of the Cloud computing environment. And it creates one object for each host among the cloud computing datacenter.

Table 4.1 shows the features (memory, function) of each agent class through the agent-based policy.

Table 4. 1 Features of each type of agents.

The Agent	*Datacenter_Coordinator*	*Host_Agent*
Memory	List of Host_Agent Objects / List of Result "Boolean"	Id / Host's Object
Function	VM Broadcasting to the Host_Agents then retrieving the results.	Check if the Host is suitable for all VM's requirements

In the context of Agent-based policy, a contract net protocol is occurring between the Datacenter_Coordinator object and Host_Agent (interleaf-agent) objects; where the VM allocation policy passes the requested VM object (VM's specifications) to Datacenter_Coordinator then it broadcasts the VM object to all Host_Agent objects (hosts) inside a datacenter (***Announcement***). Each Host_Agent verifies its host by using a special Java method in the host class called *Is_Host_Suitable* which decides either the host can host the VM or not; then Host_Agent sends back the Boolean result (True, False) to the Datacenter_Coordinator (***Bidding***). After that, the Datacenter_Coordinator put all received results into a special table and passes it to the VM allocation policy, then in ascending way the VM allocation policy selects one of the passed hosts to host the requested VM, c.f. Algorithm 4.1. Usually, in the case of failure creation to requested VM; the VM allocation policy selects the next one in the table and repeats that until the finish the table.

The algorithmic steps of the proposed Agent-based VM allocation policy, are as follows in Algorithm 4.1 [19].

Note that, the Java method *Is_Host_Suitable* is one of member methods in Host class [42], where it takes a VM object as a parameter, then return a Boolean value (True, False), after verifying host resources according to VM requirements (specifications) depends on host provisioning, allocation and scheduling algorithms without any interfere from Agent-based policy. So, Agent-based policy through using the *Is_Host_Suitable* method covers almost all the host resources; which means the Agent-based policy is full dimensional VM allocation algorithm. Also, it is a dynamic (flexible) algorithm because it depends on the current host provisioning algorithms during the VM allocation process at allocation time; hence Agent-based policy has the ability to work through heterogeneous datacenter hosts.

Therefore, the Agent-based policy does not try to create any VM in a particular host before doing a full scan to that host; that leads to prevent any failure creation attempt of VM. Consequently, there are no failure creation attempts; which means less time to consume, short allocation time and good turnaround time. All of the previous are considered as good inductions on the efficient allocation process, c.f. Chapters 5, 6 and 7.

Algorithm 4. 1. Agent-based VM Allocation Policy.

Algorithm 4.1 *Agent-based VM allocation policy*

Variables: *VM_Table Map, Boolean [] Checking*

Input: ***VM*** *Object*

Output: *Boolean* ***Result***

```
1   Result = false/ Tries = 0/ [] Checking= list of Host_Agent. Size;
2   If (VM is not already created)
3       Call the Datacenter_Coordinator
4       Datacenter_Coordinator:
5           For each (Host_Agent) // Broadcasting the VM.
6               Passing VM to Host_Agent;
7               Host_Agent:
8                   Boolean check_H;
9                   check_H = Is_Host_Suitable (VM);
10                  Return check_H (True/ False);
11              Add check_H to [] Checking;
12          End For
13          Return [] Checking;
14      Index = 0;
15      Do:
16          If (Checking [Index])
17              Get_Host (Index);
18              Result = Host.Create(VM);
19              Tries ++;
20          End If
21          If (Result)
22              Result = True;
23              Break;
24          End If
25          Else: Index ++;
26      While (! Result && Tries < Host_Agent _list. Size)
27  End If
28  Return Result;
```

The code implementation and the numerical results (Allocation Time) of Chapter 5, 6 and 7 are based on the implementation of the Agent-based VM allocation algorithm. The computational cost of this algorithm, follows, clearly, a linear pattern with respect to ***n*** (linear function) i.e.

$$F(n) = a * n + b,$$

Where ***a*** and ***b*** are constants and ***n*** is the number of hosts / VM requests. So the complexity of the Agent-based algorithm is a ***Linear Complexity*** of order ***O (n)***.

4.6 Summary

This chapter proposed the full dimensional VM allocation policy, which depends on the multi-agent system building a good awareness about datacenter resources (resources pooling). In order to make an efficient VM allocation process, the following aspects should be taken into consideration: Allocation time, resources occupied, turnaround time and VM distribution.

Next chapter presents the numerical results of the practical experiments of the proposed Agent-based VM Allocation Policy against the default policy of CloudSim toolkit, in order to assess the concept of using the multi-agent system during the allocation process. Moreover, it introduces six scenarios associated with practical experiments distributed over two categories.

Chapter 5 On the Validation of the Agent-based Policy

5.1 Introduction

After introducing the VM allocation default algorithm (policy) in Chapter 3, and Agent-based VM allocation algorithm in Chapter 4; this chapter introduces practical numerical experiments are performed through the CloudSim toolkit and distributed over six scenarios to validate the concept of the proposed algorithm (Agent-based policy).

In terms of verifying the concept of the Agent-based algorithm in this stage of the work; the obtained results from using the Agent-based algorithm through the numerical experiments are compared with the results of the CloudSim toolkit default algorithm.

The scenarios which used to verify the new VM allocation policy (algorithm) are divided into two main categories; the first one uses Time-Shared VM Scheduler [15, 17] and the second uses Space-Shared VM Scheduler [15, 18] c.f. Section 3.6. Each VM scheduler has a significant impact on VM's allocation process, based on his definition to the scheduling and sharing of the computing resources of the host between the VMs. Note that, these two VM schedulers are the default ones from the CloudSim toolkit [15, 16, 35, 36].

The objective of using the two default VM schedulers is to assess their impact on the efficiency of the VM allocation policies. Focusing on the (i) allocation time of a VM allocation policies, and (ii) number of allocated VMs (resources occupied) over the datacenter, under the specification of each VM Scheduler.

The six scenarios for verifying the Agent-based policy versus the default policy have been tested under the same conditions below, and they are divided into three scenarios use Time-Shared (Section 5.2) and another three use Space-Shared (Section 5.3) scheduler.

The following conditions and assumptions are fixed for all scenarios in any category through this chapter:

- There is one Datacenter;
- There is one Broker;
- One type of Cloudlet (application, task) is used (same features and requirements);
- A fixed number of Cloudlets that need to serve, which is double of the requested VMs.

- There is a time penalty or fine (e.g., 0.01 second) for each request for creating VM;
- Each VM creation attempt needs to 0.1 seconds regardless of the result of creation (true or false) i.e., if the allocation policy decision is not correct or not very accurate will add more time penalty.

The purpose of these experimentations is to verify the potential of the proposed Agent-based VM allocation policy for identifying and allocating a proper Datacenter's host to the requested VM. Thus, the simulation uses just one Datacenter (mono Cloud Computing), which serves one Broker (just one user). Moreover, the datacenter executes just one type of Cloudlets (applications); this is to focus on this stage of the allocation (placement) process.

Note that the adopted time penalty aims to ascertain the impact of correct decisions in both VM allocations on the allocation time of datacenter - Cloud Computing - to the Broker queries i.e., the allocation time is the measurement standard of the numerical experiments. As well as a definition of Allocation time is mentioned in Section 3.4, the technical (numerical/Mathematical) definition of Allocation time based on the proposed configuration to the CloudSim toolkit, as below:

$$At = Rq + (N * Cr) + Ak$$

Where:

At: Allocation Time, *Rq*: the VM Request Time (is fixed 0.01 second), ***N***: number of creation attempts, *Cr*: the VM Creation Time (is fixed 0.1), *Ak*: The Acknowledgement Time (is not considered 0.0 value).

So, the allocation time for requesting ***m*** VMs is:

$$\sum_{i=1}^{m} At = \sum_{i=1}^{m} [Rq + (N * Cr)) + Ak]$$

Hence, the numerical Turnaround Time definition of the same amount of VMs is as below:

$$\sum_{i=1}^{m} Tr = \sum_{i=1}^{m} [At + Re + Ex + Sb + \langle Dr + Ds \rangle]$$

Where:

Tr – Turnaround Time.

At – Allocation Time.

Re – Response Time of Cloudlet (is not considered 0.0 value).

Ex – Execution Time of Cloudlet.

Sb – Send back results Time (is not considered 0.0 value).

Dr – [Optional] VM Destroying Request Time (is not considered 0.0 value).

Ds – [Optional] VM Destroying Time (is not considered 0.0 value).

The rest of this chapter presents the scenarios through Sections 5.2 and 5.3, a discussion of the scenario's results in Section 5.4 and final Section 5.5 the chapter summary.

5.2 Time-Shared Scheduler Scenarios

As mentioned before in Section 3.6, Time-Shared scheduler its dynamic and flexible scheduler; because it distributes processing power of the host amongst VMs based on time-shared slices [15, 17]. Table 5.1 presents a brief explanation about all scenarios under this category.

Table 5. 1. Time-Shared Scheduler Scenarios Names and Objects.

NO.	***Scenario Name***	***Scenario Object***
I	Fixed Hosts vs. variable VMs requests scenario	Checking the both policies with a fixed number of hosts and variety VM's requests.
II	RAM impacting scenario	Checking the impact of increasing RAM on the performance of the both policies.
III	Hosts diversity vs. VMs diversity scenario	Testing the impact of diversity of hosts and VMs.

5.2.1 Scenario I: Fixed Hosts vs. Variable VMs Requests

The scenario I, verifies the impact of a number of VMs requested over both allocation policies (Simple, Agent-based); where the allocation time comes as a measurement over all experiments.

Conditions and assumptions: First, Using one type of VM. Second, each VM needs: one PE, 1000 MIPS, 512 MB of RAM and 1000 MHz for Bandwidth. Third, using one type of Hosts; each host contains 4 Pes (1000 MIPS) and RAM is 2048 MB. Forth, the number of hosts is fixed (50 hosts). Fifth, the number of requested VMs is variety.

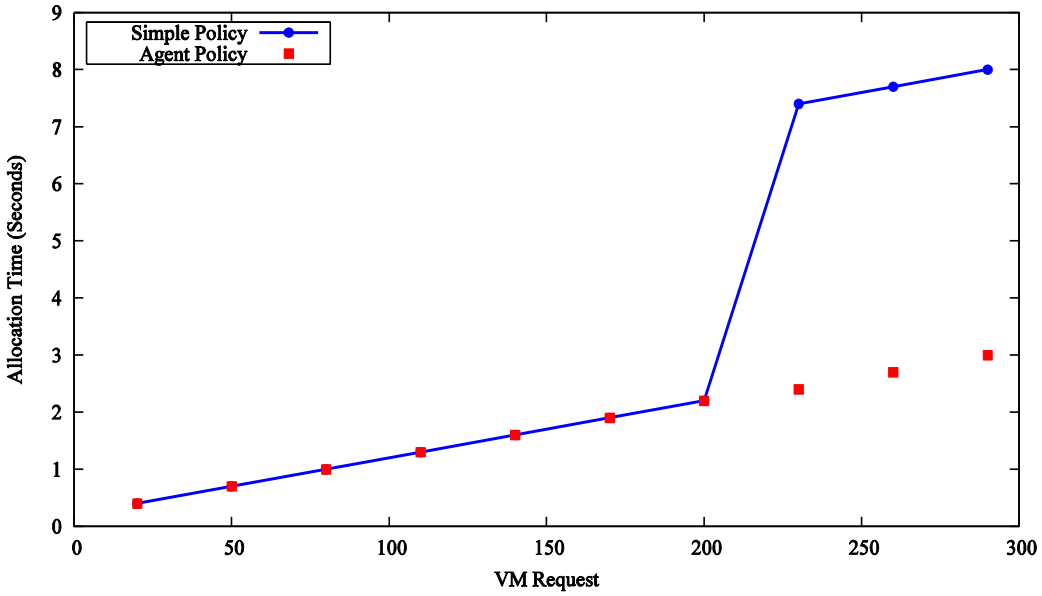

Figure 5. 1: The Allocation Time Corresponding to the Number of VMs Requests.

Based on Figure 5.1:

- The number of created VMs at all experiments same to both policies.
- VM creation takes a special order in Agent-based policy, where it's trying to allocate all available resources of host corresponds with requested VM before the move to another one.
- The allocation time for both policies it is same in under-requesting cases such as requesting 150 VMs.
- Big gap between both policies in allocation time or cloudlet starting time started in over-requesting cases such as requesting 250 VMs.
- Agent-based performs a testing for the resources (hosts) before trying to create VM; which dues to reduce allocation time.

- Based on previously, it's very clear the advantages of Agent-based policy technics over the default policy, especially in over-requesting cases.

The analyzing progress which is made by the multi-agent system in Agent-based policy, depends on the Contract Net Protocol between the Datacenter_Coordinator and Host_Agent's for all resources in the datacenter (pooling of resources). It gives the Agent-based better resources awareness over the datacenter more than the simple VM allocation policy.

Thus, the Agent-based policy takes the best decision of allocation of VM inside the host (PM). This means that the number of failed VM creation attempts is significantly reduced. In most cases, the failed attempts are completely prevented in the Agent-based policy. The datacenter, then reaches its saturation level with less number of creation attempts compared to the default policy where attempts to create VMs took place due to a weak allocation decision, based on the incomplete awareness of the Datacenter resources even when the datacenter is saturated.

The VM requesting cases of this scenario have two categories: first, the under-request category; whereas the available resources within the datacenter higher than the requesting resources for the VMs, such as a requesting of 150 VMs. Second, the over-request category; the requested resources in this category are close or equal to the available resources in the datacenter, such as a requesting of 250 VMs (Figure 5.1).

Note that, both requesting categories appeared before the saturation level of the datacenter. In the first category, both policies (Agent-based, Simple) presented the same performance efficiency for the allocation time and the allocated resources. Nevertheless, the Agent-based policy has exhibited high-performance efficiency in the second category compared with the default policy (Figure 5.1).

Consequently, when the saturation level is reached (all the datacenter resources are allocated) the Agent-based policy stops making any attempt of creating new VMs, whereas the default policy continuously attempts to create a new VM which results to many failures. Moreover, Agent-based allocates all available resources corresponding to the requested VM in ascending order (First-Pass-Fit), before moving to the next host. Aforementioned, the Agent-based VM allocation policy produces the smallest values of allocation time (Figure 5.1) especially on the over-request cases like 300 VMs.

5.2.2 Scenario II: RAM Impacting

This scenario achieves to verify the impact of increasing RAM capacity for hosts and VMs over both policies, then see how each one behaves under these conditions.

Table 5. 2. Describes the VMs Features.

Feature	*VM1*	*VM2*	*VM3*
Image size	10000 MB	20000 MB	30000 MB
RAM	1024 MB	2048 MB	3072 MB
MIPS	1000	2000	3000
Bandwidth	1000	2000	3000
Number of Pes	1	2	3
VMM	XEN	XEN	XEN

Conditions and assumptions: First, using three types of VMs as shown in table 5.2. Second, using one type of Hosts; each host contains 4 Pes (3000 MIPS) and RAM is 6144 MB. Third, the number of hosts is fixed (100 hosts). Forth, the number of requested VMs is varied, but the requesting amount of each VM type is equal.

Based on the scenario's Figures, finds:

- Figures 5.3, 5.4; the numbers of created VMs at all experiments are different to both policies.
- Figures 5.3, 5.4; VMs creation takes a special order in Agent-based policy, where it's trying to allocate all available resources of host corresponds with requested VM before the move to another one.
- According to point 2 and Figures 5.3 and 5.4; the distribution of VMs over the three types is balanced in Agent-based in the first 6 steps (up to 270 requesting).
- Figures 5.2, 5.3 and 5.4; in requesting cases (150, 180, 210, 240 and 270 VMs) Agent-based shows more efficient performance than the default policy over both the allocation time and a number of created VMs.

- Figures 5.3, 5.4; Agent-based allocated from VM type1 (small one) and VM type3 (biggest one) more than the simple one, but simple policy allocated more in type 2 (Moderate one).
- In the end, both policies occupied almost the same amount of MIPS and RAM (saturated case) during this scenario. Whereas Agent-based policy occupied less than a simple policy by around 0.67%. Both policies reached to the saturated case after same steps (690 requests).
- Figures 5.3, 5.4; the good distribution of Agent-based policy in early stages (over three types of VMs), gives the opportunity for simple policy in the last stages to allocate more from VM type2 and occupied resources more by around 0.67%.
- Figure 5.2; the allocation time for both policies it is very close in under-requesting cases such as requesting 150 VMs.
- Figures 5.2, 5.3 and 5.4; big gap between both policies in allocation time or cloudlet starting time started in this scenario in new cases such as 180 and 210 before the over-requesting cases, because the requesting amount is acceptable for Agent-based policy; this means simple policy started struggling before another one over time and resources occupied.

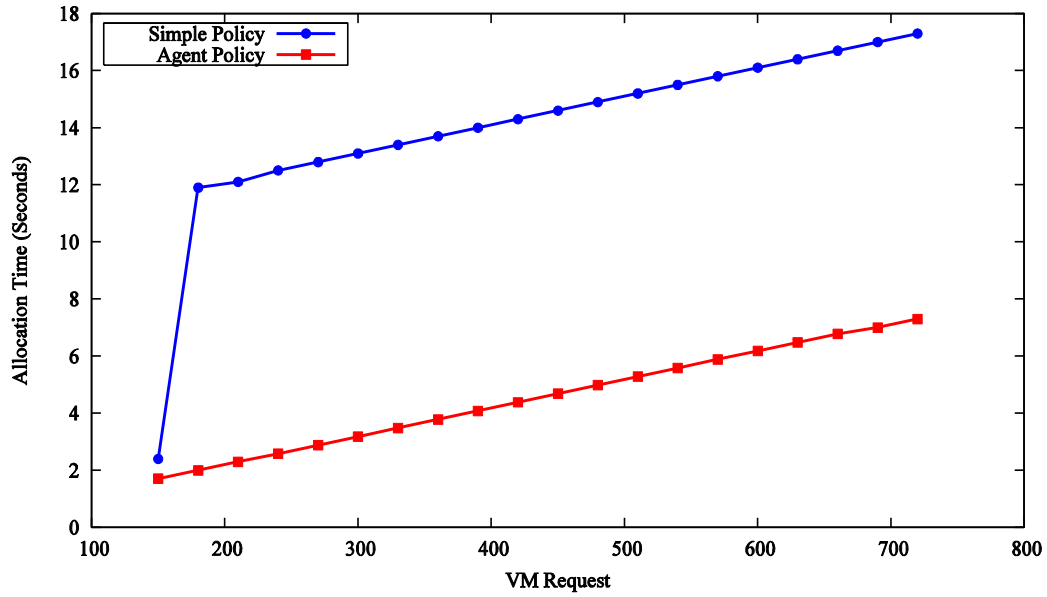

Figure 5. 2: The Allocation Time Corresponding to the Number of VMs Requests (3 types).

- Figure 5.2; the gap is continuing across the over-requesting cases such as 270 VMs.
- Agent-based performs a testing for the resources (hosts) before trying to creation VM; which dues to reduce the allocation time (reducing allocation time), Figure 5.2.
- The agent-based policy works efficiently over VM's diversity.
- Based on previously, it's very clear the advantages of using a multi-agent system on the Agent-based VM allocation policy performance over the default policy especially in over-requesting cases.

Practically, the checking operation which is done by the multi-agent system (agent-based VM allocation policy) depends on a Contract Net Protocol between their entire agents for all resources in the datacenter (pooling of resources). It gives the Agent-based policy better resources awareness over the datacenter more than the simple policy.

Consequently, the Agent-based policy takes the best decision of allocation of VM inside the host (PM). This means the number of failed VM creation attempts is significantly reduced. In most cases, the failed attempts are completely prevented in the Agent-based policy. Hence, the datacenter reaches to its saturation level with less number of creation attempts compared to the default policy, where the attempts to create VMs took place due to a weak allocation decision, based on the incomplete awareness of the Datacenter resources even when the datacenter is saturated c.f. Figure 5.2.

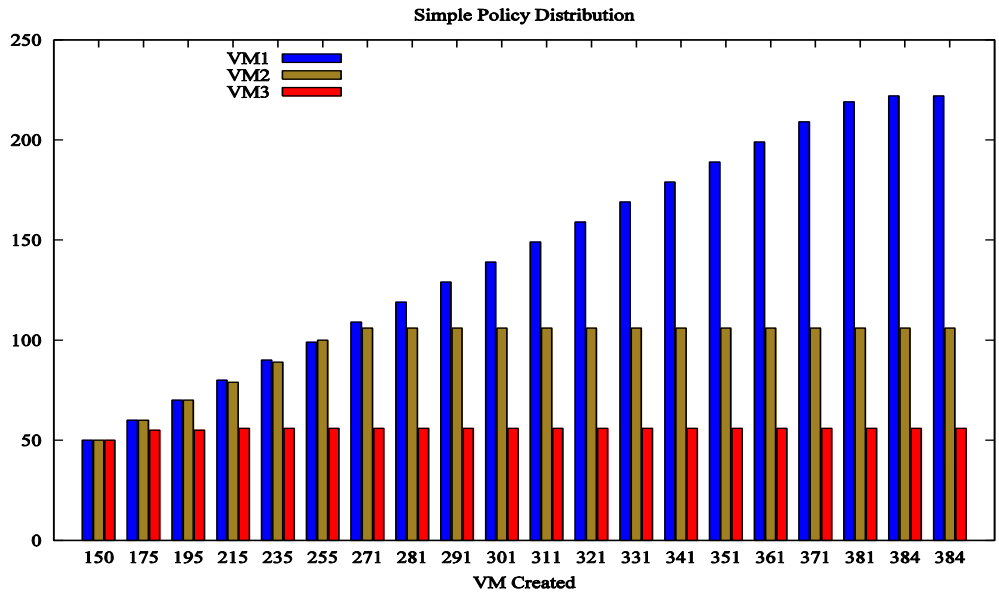

Figure 5. 3: The Distribution of Created VMs in Simple Policy.

The VM requesting have two categories: first, the under-request category; whereas the available resources within the datacenter higher than the requesting resources for the VMs, such as a requesting of 160 VMs or less. Second, the over-request category; the requested resources in this category are close or equal to the available resources in the datacenter, such as a requesting of 600 VMs c.f. Figure 5.2.

The both requesting categories showed up before the saturation level of the datacenter. Where, in the first category both allocation policies (Agent-based, Simple) presented the same performance efficiency for the allocation time and allocated resources. Nevertheless, the Agent-based policy has exhibited high-performance efficiency in the second category compared with the default policy c.f. Figure 5.2.

Note that, the under-request category has been shrunk through the scenario's experiments compared with scenario 1 (Section 5.2.1). That means the increasing of RAM gives the Agent-based an opportunity to perform in an efficient way more than the default policy.

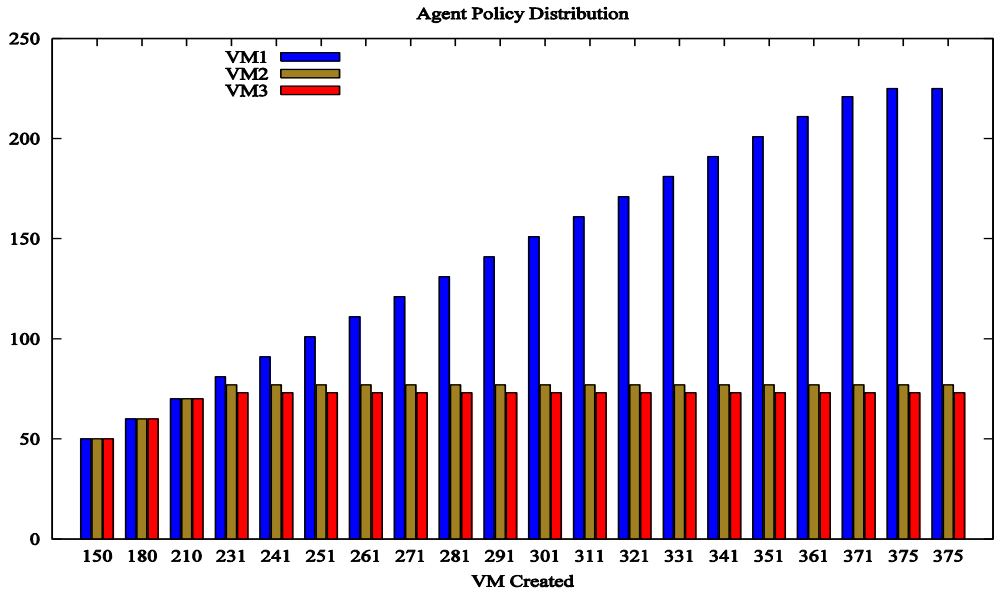

Figure 5. 4: The Distribution of Created VMs in Agent-Based Policy.

Consequently, when the saturation level is reached, again the Agent-based policy stops making any attempt for creating new VMs, whereas the default policy continuously attempts to create a new VM which leads to many failures.

Additionally, Agent-based policy allocates all available resources corresponding to the requested VM in ascending order (First-Pass-Fit), before moving to the next host. That gives a well-balanced distribution between the VM types amongst the hosts of datacenter through the scenario's numerical experiments c.f. Figures 5.2, 5.3 and 5.4.

Thus, the Agent-based VM allocation policy produces the smallest values of allocation time c.f. Figure 5.2 especially in the over-request cases such as a 750 VMs.

5.2.3 Scenario III: Hosts Diversity vs. VMs Diversity

This scenario aims to verify the impact of diversity of hosts and VMs on behaving of both allocation policies (simple, Agent-based); where it has three types of the host with different features and three types of VMs as a previous scenario (Sub-Section).

The scenario's conditions and assumptions: First, using three types of VMs as shown in table 5.4. Second, using three types of Hosts as shown in table 5.3, Third, the number of hosts is fixed (120 hosts); divided into three groups equally, 40 hosts for each type. Forth, the number of requested VMs is varied, but the requesting amount of each VM type is equal.

Table 5. 3. Host's Types.

Host 1	***Host 2***	***Host 3***
4 Pes (3000 each)	2 Pes (3000 each)	3 Pes (3000 each)
RAM: 6144 MB	RAM: 4096 MB	RAM: 2048 MB

Based on the scenario's Figures:

- Figures 5.6, 5.7; the numbers of created VMs at all experiments are different to both policies.
- Figures 5.6, 5.7; VMs creation takes a special order in Agent-based policy, where it's trying to allocate all available resources of host corresponds with requested VM before the move to another one.
- According to point 2 and Figures 5.6 and 5.7, the distribution of VMs over the three types is balanced in Agent-based policy in the first 5 steps (up to 180 requesting).

- Figures 5.5, 5.6 and 5.7; in requesting cases (60 to 510 VMs) Agent-based shows more efficient performance than the default policy over both the allocation time and a number of created VMs.
- Figures 5.6, 5.7; Agent-based policy allocated from VM type2 (moderate one) and VM type3 (biggest one) more than the simple one, but simple policy allocated more in type 1 (smallest one).

Table 5. 4. The VMs Features.

Feature	*VM1*	*VM2*	*VM3*
Image size	10000 MB	20000 MB	30000 MB
RAM	1024 MB	2048 MB	3072 MB
MIPS	1000	2000	3000
Bandwidth	1000	2000	3000
Number of Pes	1	2	3
VMM	XEN	XEN	XEN

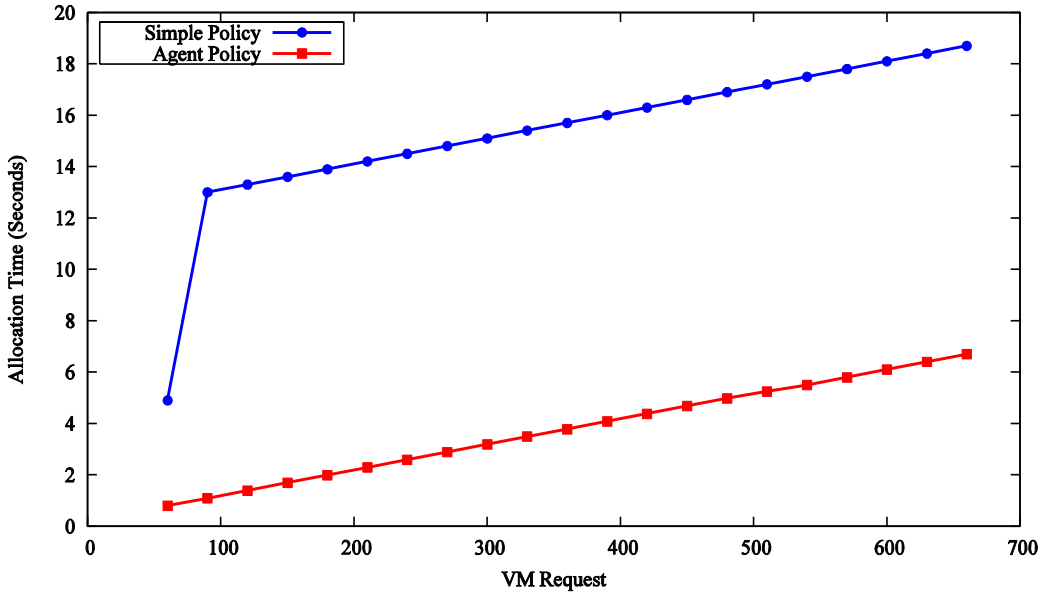

Figure 5. 5: The Allocation Time Corresponding to the Number of VMs Requests (3 types).

- In the end, both policies occupied the same amount of MIPS and RAM (saturated case) during this scenario. But, Agent-based reached to the saturated case after 16 steps (510 VMs requests), c.f. Figure 5.7; regard to simple policy takes 20 steps (630 VMs requests) to get same level, c.f. Figure 5.6.

- Figure 5.5; the allocation time for both policies it is a bit close in under-requesting cases such as requesting 60 VMs.
- Figures 5.5; big gap between both policies in allocation time started in this scenario –same as previous one- in new cases such as 90 before the over-requesting cases, due to the requesting amount is acceptable for Agent-based policy; this means simple policy started struggling before another one over time and resources occupied.
- Figure 5.5; the gap is continuing across the over-requesting cases starting from 120 VMs.
- Figure 5.5; Agent-based performs testing for the resources (hosts) before trying to creation VM; which dues to reduce allocation time and show high efficiency over VM's and Hosts diversity.
- According to that, it's very clear the advantages of Agent-based policy technics over the default policy, especially in over-requesting cases.

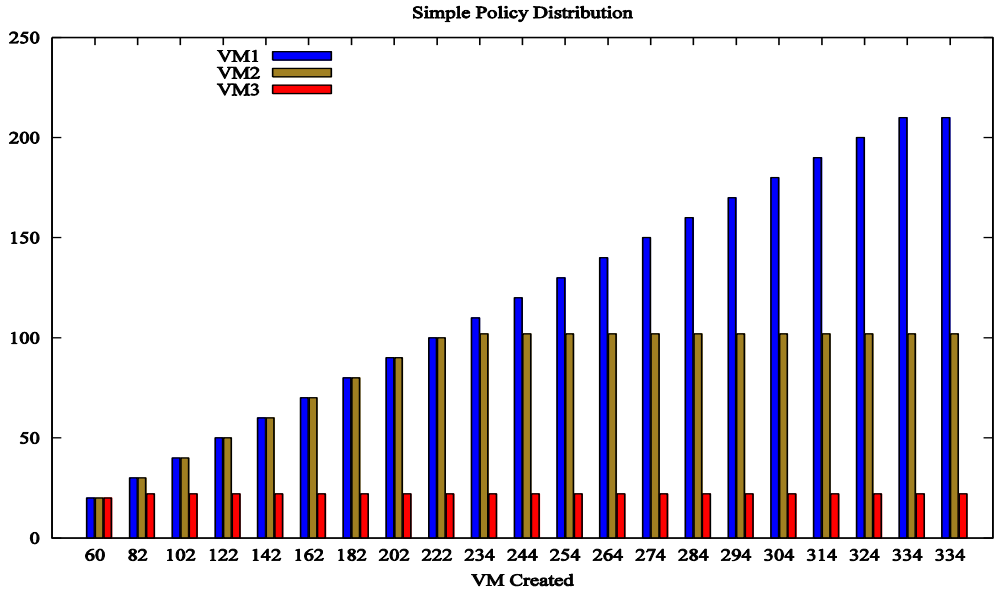

Figure 5. 6: The Distribution of Created VMs in Simple Policy.

Technically, the parsing operation which is performed by the multi-agent system in agent-based policy depends on the Contract Net Protocol between a Datacenter_Coordinator and Host_Agents for all resources in the datacenter (pooling of resources). It gives the Agent-based policy a good resources awareness over the datacenter compares with the simple VM allocation policy.

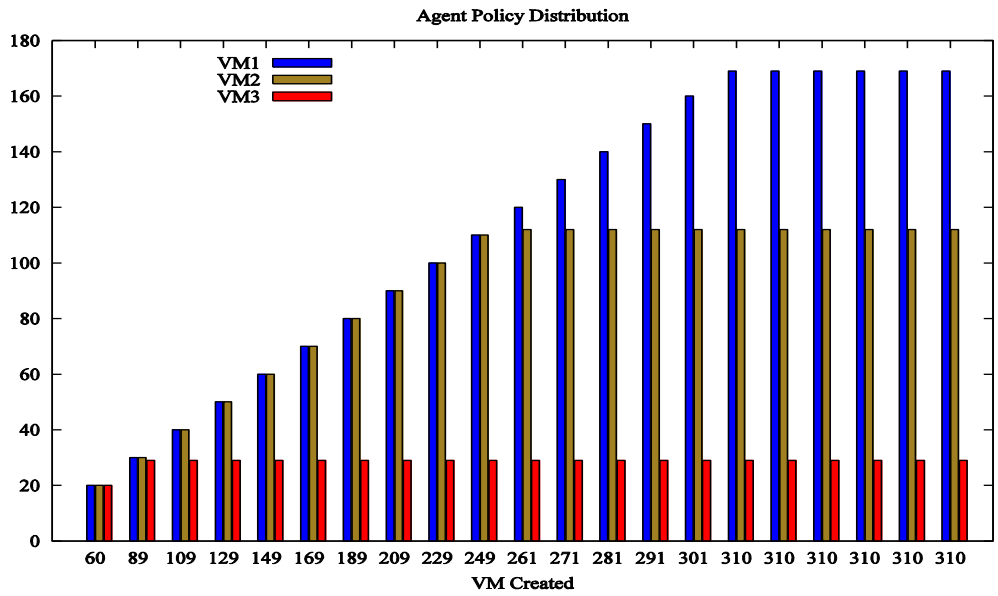

Figure 5. 7: The Distribution of Created VMs in Agent-Based Policy.

Therefore, the Agent-based policy takes the best decision of allocation of VM inside the host (PM). This means that the number of failed VM creation attempts is significantly reduced, or in most cases, the failed attempts are completely prevented in the Agent-based policy. Furthermore, the datacenter reaches its saturation level with less number of creation attempts compared to the default policy, where the attempts to create VMs took place due to a weak allocation decision, based on the incomplete awareness of the Datacenter resources even when the datacenter is saturated c.f. Figure 5.5.

The VM requesting cases again have two categories: first, the under-request category; whereas the available resources within the datacenter higher than the requesting resources for the VMs, such as a requesting of 50 VMs or less. Second, the over-request category; the requested resources in this category are close or equal to the available resources in the datacenter, such as a requesting of 650 VMs c.f. Figure 5.5.

Both requesting categories appeared before the saturation level of the datacenter. In the first category, both allocation policies (Agent-based, Simple) presented the same performance efficiency for the allocation time and allocated resources. Nevertheless, the Agent-based policy has exhibited high-performance efficiency in the second category compared with the default policy c.f. Figure 5.5.

The under-request category shrunken, even almost it's disappeared through the scenario's experiments compared to previously scenarios. The increasing of RAM and the host types diversity, give the Agent-based policy an opportunity to perform in an efficient way more than the default policy under those conditions.

Consequently, when the saturation level is reached, an Agent-based policy stops making any attempt of creating new VMs, whereas the default policy continuously attempts to create a new VM which results in many failures. Thus, the Agent-based policy produces the smallest values of allocation time c.f. Figure 5.5 especially in the over-request cases such as a 700 VMs.

Eventually, Agent-based policy allocates all available resources corresponding to the requested VM in ascending order (First-Pass-Fit), before moving to the next host. Which gives a well-balanced distribution between the VM types among the hosts of datacenter through the scenario's numerical experiments c.f. Figures 5.5, 5.6 and 5.7.

5.3 Space-Shared Scheduler Scenarios

This section presents the scenarios of Space-Shared scheduler (Section 3.6); where it is contradictory with the Time-Shared scheduler because it is considered as a static or inflexible scheduler of VMs [15, 18]. More specifically, it assigns host cores to a particular VM and prevents any sharing of these cores with other VMs through the lifecycle of current VM, which means the number of created VMs by using Space-Shared less than using Time-Shared scheduler at same scenario conditions; as mentioned in Section 3.6.

Table 5.5 gives an overview of all scenarios in this section under the Space-Shared category.

Table 5. 5. Space-Shared Scheduler Scenarios Names and Objects.

NO.	***Scenario Name***	***Scenario Object***
I	Hosts number vs. VMs requests scenario	Verifying the performance the both policies with increasing the number of hosts and VM's requests
II	Impacting of VMs variety scenario	Verifying the impact of VM's variety on both policies performance
III	Hosts diversity vs. VMs diversity scenario	Testing the impact of diversity of hosts and VMs.

5.3.1 Scenario I: Hosts Number vs. VMs Requests

This sub-section (scenario) aims to test the efficiency of both VM allocation policies (Simple, Agent-based) under special conditions and assumptions (below); where the allocation time of VM allocation policies vs number of hosts in the datacenter (c.f. Figure 5.8) is taken as a measurement for scenario's experiments.

The scenario's conditions and assumptions: First, using same VM type (same features and requirements). Second, using two types of Hosts; which both types have the same features apart, the number of processors – PE's – where one of them has a double number of processors and MIPS of each PE are 1000 and Ram is 2048 MB. Third, the same number of each host's type (50% for each one). Forth, the number of requested VMs 10 times more than the number of hosts. Such as 10 hosts the broker requests 100 VMs.

Based on Figure 5.8, bellows points are observed:

- The number of created VMs at all experiments same to both policies.
- VM creation takes a special order in Agent-based policy, where it's trying to allocate all available resources of host corresponds with requested VM before the move to another one.
- Based on the previous conditions the allocation time or cloudlet starting time in our policy is less than the default policy by around 50%.
- All extreme boundaries cases of host types, such as 10%: 90%, 90%: 10% and 70%: 30 are tried. The experiments show no impact on results where all experiments gave the same results.
- Based on the previous point, the allocation time and a number of created VMs depending on the number of hosts (resources pool) and the number of requests to create VMs.

Parsing progress which is performed by the multi-agent system in Agent-based depends on the Contract Net Protocol between the Datacenter_Coordinator and Host_Agents for all resources in the datacenter (pooling of resources). It gives the Agent-based better resources awareness over the datacenter more than the simple policy.

Therefore, Agent-based takes the best allocation decision of VM inside the host. This means that the number of failed VM creation attempts is significantly reduced. According to that, in most cases, the failed attempts are completely prevented in the Agent-based policy.

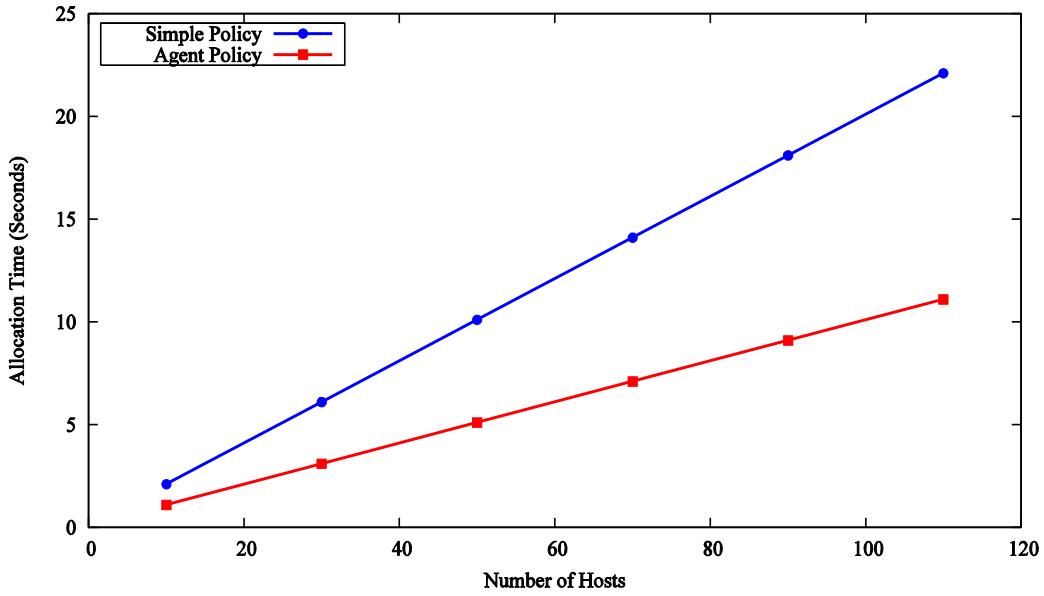

Figure 5. 8: The Allocation Time of Both Policies with the Number of Hosts.

The datacenter, then reaches its saturation level with less number of creation attempts compared to the default policy, where attempts to create VMs took place due to a weak allocation decision, based on the incomplete awareness of the Datacenter resources even when the datacenter is saturated.

Consequently, when the saturation level is reached the Agent-based policy stops making any attempt of creating new VMs, whereas the default policy continuously attempts to create a new VM which leads to many failures. Moreover, Agent-based policy allocates all available resources corresponding to the requested VM in ascending order (First-Pass-Fit), before moving to the next host. Thus, the Agent-based produces the smallest values of allocation time (c.f. Figure 5.8).

5.3.2 Scenario II: Impacting of VMs Variety

This sub-section (scenario) purposes to test the efficiency of both allocation policies (Agent-based, Simple) under the following special conditions and assumptions:

The scenario's conditions and assumptions: First, using three types of VMs as shown in table 5.6. Second, using one type of Hosts; each host contains 4 Pes (3000 MIPS) and RAM is 2048

MB. Third, the number of hosts is fixed (100 hosts). Forth, the number of requested VMs is varied, but the requesting amount of each VM type is equal.

Table 5. 6. The VMs Features.

Feature	*VM1*	*VM2*	*VM3*
Image size	10000 MB	20000 MB	30000 MB
RAM	512 MB	1024 MB	1536 MB
MIPS	1000	2000	3000
Bandwidth	1000	2000	3000
Number of PEs	1	2	3
VMM	XEN	XEN	XEN

Based on the scenario's Figures; below are observed:

- Figures 5.10, 5.11; the numbers of created VMs of all experiments are different to both policies, but in the end, both policies allocate the same number of VMs.
- Figures 5.10, 5.11; VM creation takes a special order in Agent-based, where it's trying to allocate all available resources of host corresponds with requested VM before the move to another one.
- According to point 2 and Figures 5.10 and 5.11, the distribution of VMs among the three types is balanced in Agent-based policy.
- Figures 5.9, 5.10 and 5.11; in requesting cases (120,150, 210 and 240 VMs) Agent-based shows more efficient performance than the default policy over both the allocation time and a number of created VMs.
- In the end, both policies have not occupied the same amount of MIPS and RAM (to reach the saturation level of datacenter) during this scenario. Whereas Agent-based occupied more than Simple one by around 12.3%. Moreover, Agent-based reached to the saturated case after 4 steps (210 VMs requests), c.f. Figure 5.11; regard to simple policy takes 6 steps (270 VMs requests) to get same level, c.f. Figure 5.10.
- Figure 5.9; the allocation time for both policies it is same in under-request cases such as a requesting 120 VMs.

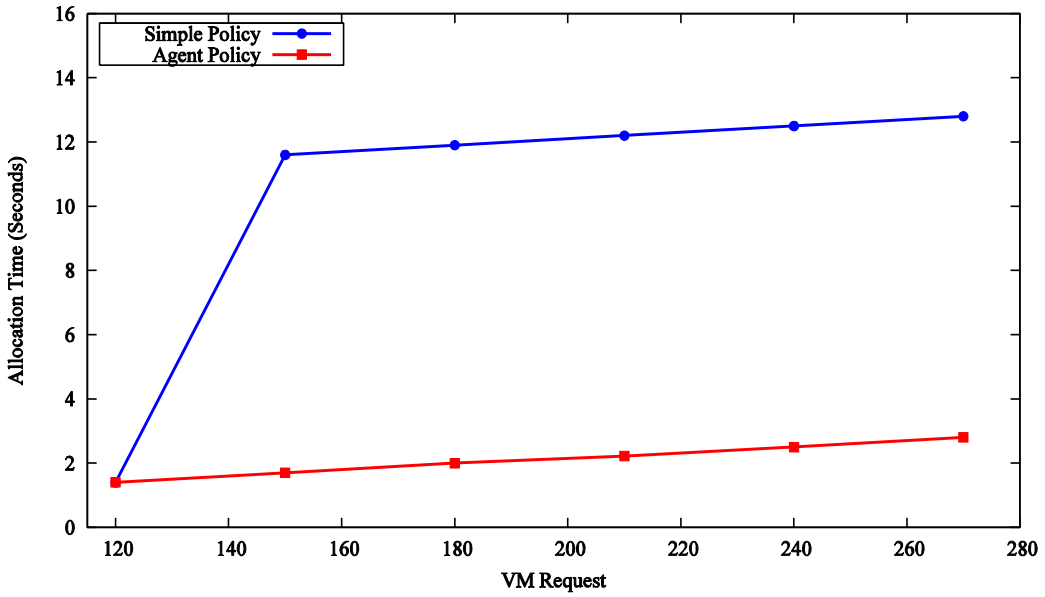

Figure 5. 9: The Allocation Time Corresponding to the Number of VMs Requests (3 types).

- Figures 5.9, 5.10 and 5.11; big gap between both policies in the allocation time started in this scenario in the new case, such as 150 and 180 before the over-requesting cases. Because the requesting amount is acceptable for Agent-based. Which means a simple policy started struggling before Agent-based on the allocation time and occupied resources.
- Figure 5.9; the gap is continuing across the over-requesting cases such as 240 VMs.
- Figures 5.9, 5.10 and 5.11; Agent-based performs testing to the resources (hosts), before trying to create any VM. Thus the allocation time is reduced, and the datacenter does not need to a large number of requested VMs to reach to the saturation level.
- The agent-based policy works efficiently over VM's diversity.
- Consequently, it's very clear the advantages of Agent-based techniques compared with the default policy, especially among the over-request cases.
- Compared with the Time-Shared scheduler under the same conditions, the number of created VMs is more than this scenario; because the Space-Shared scheduler is more static.

Consequence these observations, the parsing progress, which is performed by the multi-agent system Agent-based policy depends on the Contract Net Protocol technique, for all resources in the datacenter (pooling of resources). It gives the Agent-based better resources awareness over the datacenter more than the simple policy.

Therefore, Agent-based takes the best allocation decision of VM inside the host. Which means that the number of failed VM creation attempts is significantly reduced. In most cases, the failed attempts are completely prevented in Agent-based policy.

The datacenter by using Agent-based policy reaches to the saturation level with less number of creation attempts compared to the default policy, where attempts to create VMs took place due to a weak allocation decision, based on the incomplete awareness of the Datacenter resources even when the datacenter is saturated c.f. Figure 5.9.

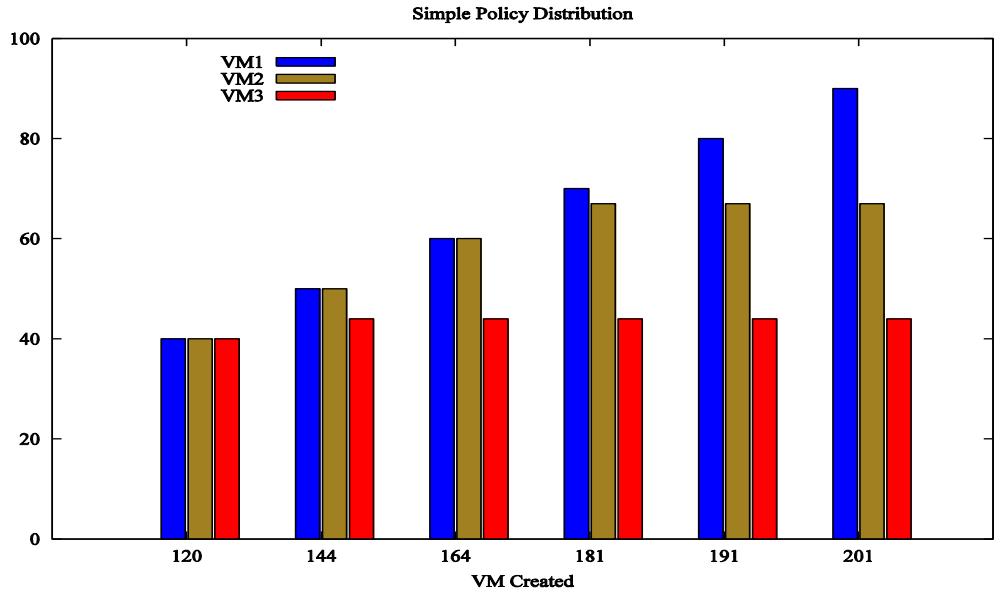

Figure 5. 10: The Distribution of Created VMs in Simple Policy.

The VM requesting cases of this scenario have two categories: first, the under-request category; whereas the available resources within the datacenter higher than the requesting resources for the VMs, such as a requesting of 120 VMs. Second, the over-request category; the requested resources in this category are close or equal to the available resources in the datacenter, such as a requesting of 240 VMs c.f. Figure 5.9.

Note that, the both requesting categories appeared before the saturation level of the datacenter. In the first category, both policies (Agent-based, Simple) presented the same performance efficiency for the allocation time and allocated resources. Nevertheless, the Agent-based has exhibited high-performance efficiency in the second category compared with the default policy c.f. Figure 5.9. Thus, the Agent-based policy produces the smallest

values of allocation time c.f. Figure 5.9 especially on the critical (over-request) cases such as a 200 VMs.

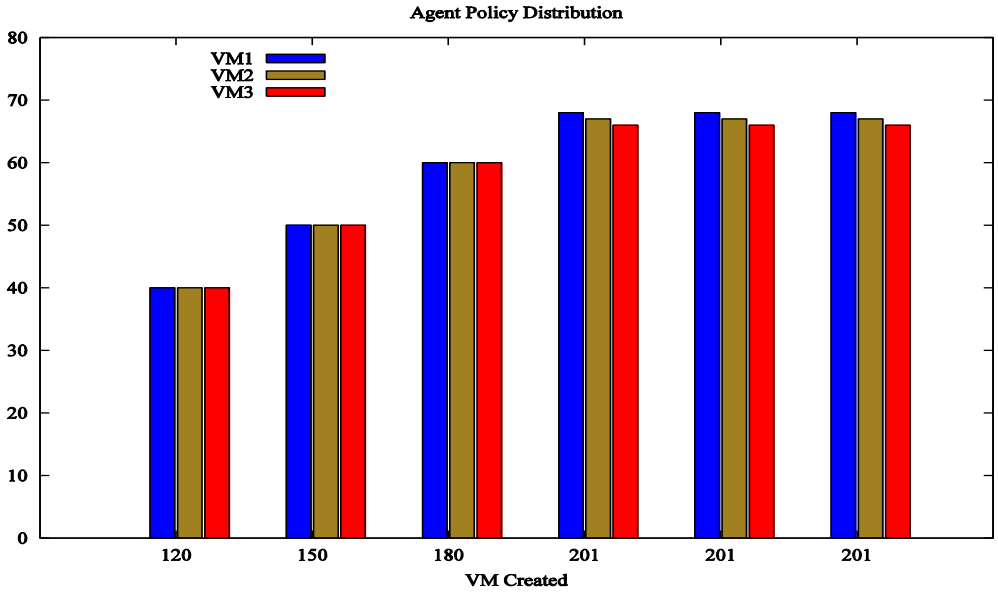

Figure 5. 11: The Distribution of Created VMs in Agent-Based Policy.

Consequently, when the saturation level of the datacenter is reached, the Agent-based stops making any attempt of creating new VMs, while the default policy continuously attempts to create a new VM, which results many failures. Moreover, Agent-based allocates all available resources corresponding to the requested VM in ascending order (First-Pass-Fit), before moving to the next host. That gives a well-balanced distribution between the VM types amongst the hosts of datacenter through the scenario's numerical experiments c.f. Figures 5.9, 5.10 and 5.11.

5.3.3 Scenario III: Hosts Diversity vs. VMs Diversity

This scenario (sub-section) aims to verify the impact of diversity of hosts and VMs on behaving of both allocation policies (simple, Agent-based); where this scenario has three types of the host with different features and three types of VMs as a previous scenario.

Table 5. 7. Shows the difference between host's types.

Host 1	*Host 2*	*Host 3*
4 Pes (3000 each) RAM: 6144 MB	2 Pes (3000 each) RAM: 4096 MB	3 Pes (3000 each) RAM: 2048 MB

The scenario's conditions and assumptions: First, using three types of VMs as shown in table 5.8. Second, using three types of Hosts as shown in table 5.7, Third, the number of hosts is fixed (120 hosts); divided into three group equally, 40 hosts for each type. Forth, the number of requested VMs is varied, but the requesting amount of each VM type is equal.

Table 5. 8. Describes The VMs Features.

Feature	*VM1*	*VM2*	*VM3*
Image size	10000 MB	20000 MB	30000 MB
RAM	1024 MB	2048 MB	3072 MB
MIPS	1000	2000	3000
Bandwidth	1000	2000	3000
Number of Pes	1	2	3
VMM	XEN	XEN	XEN

Based on the scenario's Figures; finds:

- Figures 5.13, 5.14; the numbers of created VMs at all experiments are different to both policies.
- Figures 5.13, 5.14; VM creation takes a special order in Agent-based, where it's trying to allocate all available resources of host corresponds with requested VM before the move to another one.
- According to point 2 and Figures 5.13 and 5.14, the distribution of VMs over the three types is balanced by using Agent-based in the first 4 steps (up to 120 requesting), moreover the distribution for VM type1 and type 2 balancing over all stages.

- Figures 5.12, 5.13 and 5.14; during requesting cases from 30 to 300 VMs, Agent-based exhibits more efficient performance than the default policy over both the allocation time and a number of created VMs.

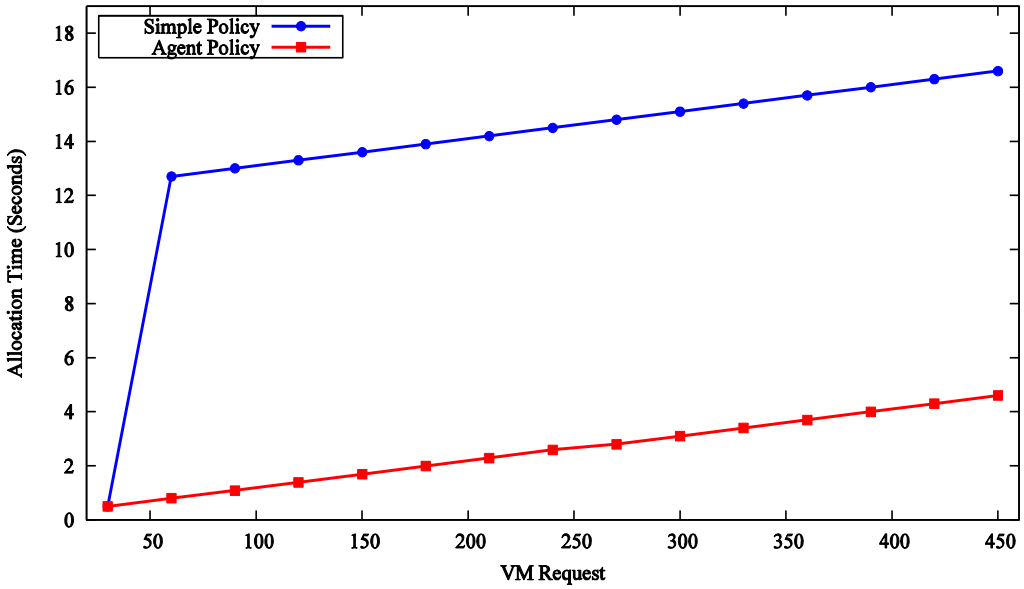

Figure 5. 12: The Allocation Time Corresponding to the Number of VMs Requests (3 types).

- Figures 5.13, 5.14; Agent-based allocated from VM type2 (moderate one) and VM type3 (biggest one) more than the simple one, but simple policy allocated more in type 1 (smallest one).
- Figures 5.13, 5.14; in the end, both policies occupied the same amount of MIPS and RAM (saturated case) during this scenario. But Agent-based reached to the saturated case after 9 steps (270 VMs requests); regard to simple policy takes 14 steps (420 VMs requests) to get the same level.
- Figure 5.12; the allocation time for both policies it is same in under-requesting cases such as requesting 30 VMs or less.
- Figures 5.12, 5.13 and 5.14; Big gap between both policies in allocation time started in this scenario –same as previous one- in new case such as 60 and 90 before the over-requesting cases, due to the requesting amount is acceptable for Agent-based policy; this means simple policy started struggling early over time and resources occupied.
- Figure 5.12; the gap is continuing across the over-requesting cases starting from 120 VMs.

- Figure 5.12; Agent-based performs testing for the resources (hosts) before trying to creation VM; which dues to reduce allocation time.
- The agent-based policy works efficiently over VM's and Hosts diversity.
- Based on previous, it's very clear the advantages of Agent-based policy technics over the default policy especially in the critical (over-requesting) cases.

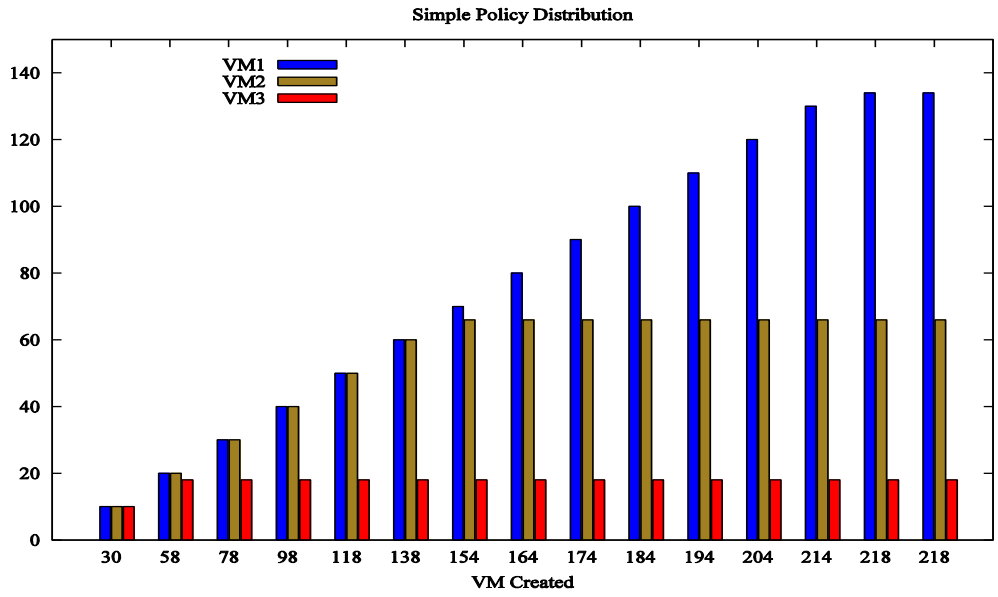

Figure 5. 13: The Distribution of Created VMs in Simple Policy.

Particularly, the parsing operation which is performed by the multi-agent system through Agent-based policy, for the datacenter resources (pooling of resources). It gives the Agent-based policy a good resources awareness over the datacenter more than the simple policy.

According to that, Agent-based takes the best allocation decision of VM inside the host (PM). This means that the number of failed VM creation attempts is significantly reduced. Further, in most cases, the failed attempts are completely prevented in the Agent-based. And the datacenter reaches to the saturation level through less number of creation attempts compared to the default policy, because it has an incomplete awareness of the Datacenter resources c.f. Figure 5.12.

There are two categories for the VM requesting cases in this scenario: first, the under-request category; whereas the available resources within the datacenter higher than the requesting resources for the VMs, such as a requesting of 30 VMs or less. Second, the over-request

category; the requested resources in this category are close or equal to the available resources in the datacenter, such as a requesting of 400 VMs c.f. Figure 5.12.

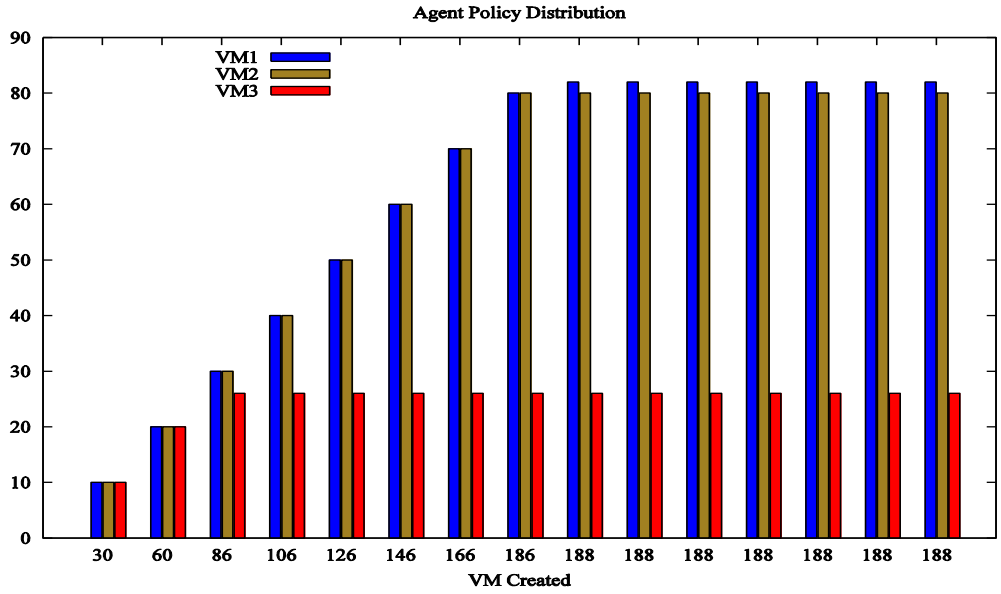

Figure 5. 14: The Distribution of Created VMs in Agent-Based Policy.

In the first category, both policies (Agent-based, Simple) presented the same performance efficiency for the allocation time and allocated resources. Nevertheless, the Agent-based policy has exhibited high-performance efficiency in the second category compared with the default policy c.f. Figure 5.12. Note that, both categories showed up before the saturation level of the datacenter.

The under-request category shrank even almost it disappeared through the scenario's experiments compared with scenarios 2. The increasing of RAM and the host types diversity, give the Agent-based an opportunity to perform in an efficient way more than the default policy under those conditions.

Consequently, when the saturation level is reached, Agent-based stops making any attempt of creating new VMs, while the default policy continuously attempts to create a new VM which leads to many failures. Moreover, Agent-based allocates all available resources corresponding to the requested VM in ascending order (First-Pass-Fit), before moving to the next host. That

gives a well-balanced distribution between the VM types amongst the hosts of datacenter through the scenario's numerical experiments c.f. Figures 5.12, 5.13 and 5.14.

Thus finally, the Agent-based policy produces the smallest values of allocation time c.f. Figure 5.12 especially on the over-request cases such as a 450 VMs.

5.4 The Discussion

This chapter presented three scenarios for each of two categories aiming to validate the use of the multi-agent system during the allocation process, through verifying and testing the performance and potential of the proposed Agent-based policy in the allocation amongst the Cloud datacenter resources. In addition, Agent-based results were compared with those of the default policy of CloudSim toolkit.

The scenarios cover a wide range of different cases, which led to proper judgment about the potential of using a multi-agent system through the Agent-based policy for allocating VMs in the cloud datacenter. Tables 5.9 and 5.10 presented fast easy reviews for all scenarios and the results which have been achieved during the scenarios experiments. Note that, both tables show that proposed Agent-based policy achieved very good results and improved the allocation process through all scenarios.

The use of the Space-Shared scheduler among the hosts reduced the number of created VMs inside the Datacenter compared with the number of created VMs by using the Time-Shared scheduler. The reduction of created VMs took place over using both VM allocation policies (Simple, Agent-based).

The proposed Agent-based policy for VM allocation over datacenter, which was implemented using the CloudSim toolkit and based on numerical experiments, shows a highly efficient performance in terms of the Allocation Time and the amount of Occupied Resources (c.f., VMs). Moreover, this policy in diversity cases gives a balanced distribution between different VM types over all hosts in datacenter i.e., this allocation policy performs efficiently under a variety of demands from brokers or users for a variety of cloudlet requirements.

Table 5. 9. Presents a Brief Review of Scenarios in Time-Shared Category.

Time – Shared Category				
No.	***Allocation Time***	***resources occupied***	***VMs Distribution***	***Saturation Case***
I	Agent-based policy better than Simple policy by 50% in the over requesting cases	Same amount	-	-
II	Agent-based policy better than Simple policy by 50% in the over requesting cases and overall	Almost the both occupied the same amount, agent-based less than simple by 0.67%.	Agent-based policy balanced more than Simple policy for the 3 types	Agent-based policy reached before Simple policy
III	Agent-based policy better than Simple policy by 50% in the over requesting cases and overall	Same amount	Agent-based policy balanced more than Simple policy for the 3 types	Agent-based policy reached before Simple policy

Specifically, the Agent-based showed better performance over both Time-Shared and particularly, Space-Shared VM Schedulers in terms of allocation time and VM resource allocation. Notably, the final number of created VMs and the amount of occupied resources for both policies (simple, agent-based) depends on which VM scheduler is used, the variety of VMs types and diversity of host types (c.f., physical machines variety). The new Agent-based policy reduced the allocation time over all experiments in all scenarios up to 50% over the toolkit's default simple policy and thus, it improved the allocation time of VM allocation of datacenter and the overall performance of the datacenter (c.f., Cloud computing core).

Table 5. 10. Presents a Brief Review of Scenarios in Space-Shared Category.

Space – Shared Category				
No.	***Allocation Time***	***resources occupied***	***VMs Distribution***	***Saturation Case***
I	Agent-based policy better than Simple policy by 50%	Same amount	-	-
II	Agent-based policy better than Simple policy by 50% in the over requesting cases	Agent-based occupied more than the Simple policy by 12.3%.	Agent-based policy better than Simple policy for the 3 types	Agent-based policy reached before Simple policy
III	Agent-based policy better than Simple policy by 50% in the over requesting cases and overall	Same amount	Agent-based policy balanced more than Simple policy for the 3 types	Agent-based policy reached before Simple policy

Using a multi-agent system technology gave the Agent-based policy high potential to build a good awareness about the datacenter resources; in this way, the proposed Agent-based policy achieved the best VM allocation decision. Consequently, reducing the number of VM creation attempts (succeed, failed) led to reducing the allocation time according to its definition (c.f. Section 5.1). This means an improvement of the allocation process among the Cloud datacenter.

Finally, the validation of the multi-agent system during the allocation process demonstrated the superiority of the proposed Agent-based policy over the default policy of CloudSim toolkit.

The Agent-based policy is compared with real allocation algorithms in the state of the art during the following two chapters (Chapters 6 and 7).

5.5 Summary

This chapter conducted practical experiments in order to validate the concept of using a multi-agent system, which was based on a collection of numerical results associated with six scenarios, which were divided into two categories, namely a Time-Shared scheduler and a Space-Shared scheduler. Moreover, six scenarios were employed to cover several different cases of allocation process between the broker and datacenter. This gave sufficient validation about the superiority of the proposed Agent-based policy performance against the default policy of CloudSim toolkit.

In Chapter 6, the proposed Agent-based policy is compared with four of the state of the art algorithms for VM allocation amongst the Cloud datacenter.

Chapter 6 The Agent-based Policy VS. Four of the state of the art Policies: One-Dimensional Comparisons

6.1 Introduction

The comparative analysis approach for verifying the results of the new proposed protocol, algorithm, solution, methodology… etc. with the current state of the art (real-life) algorithms, protocols... etc... It is considered one of the most significant research approaches to evaluate the new contributions of any research work in any researching field such as a Cloud computing. Moreover, the comparative approach gives induction on the correlation with surrounding research area, the amount of improvement and the applicable possibility.

The concept of using a multi-agent system in the allocation process (Agent-based policy) is approved in Chapter 5 through broad numerical experiments over varying scenarios. Thus, this chapter presents the comparison between Agent-based policy and four of the state of the art algorithms to prove the potential of Agent-based results and improvement against the real-life algorithms.

Note that, the Agent-based policy in this chapter, it has the same implementation, configuration, and design which used in Chapter 4 without any modification or developing. So that, it still has simple testing/verifying criteria (First-Pass-Fit) and it consists of simple make-decision agents' c.f. Section 4.5.

In the context of this chapter, four conventional algorithms are implemented and configured according to the CloudSim toolkit environment specification c.f. next section; in order to compare with Agent-based algorithm among different 6 scenarios, which are divided into two categories: (i) Time-Shared scheduler [17] and (ii) Space-Shared scheduler [18]. Furthermore, the comparisons through varying scenarios aim to show up the advantages and disadvantages of four algorithms and Agent-based algorithm, and it attempts to expose the novelty and originality of the proposed algorithm over current conventional algorithms.

Most of the current state of the art algorithms are conventional Bin-Packing heuristic algorithms [13, 75] such as BF (Best-Fit), BFD (Best-Fit-Decreasing), FT (First-Fit), FTD (First-Fit-Decreasing), …etc.. Moreover, the decreasing algorithms like FTD, need to pre-knowledge about the requested VMs before starting the allocation process to perform

descending (decreasing) sorting to all VMs, then allocate the VMs corresponding to the sorting order. Thus, no decreasing algorithm is used through the comparison Chapters (6, 7); due to in most real-life cases there is no pre-knowledge about the requested VMs, and the configuration of CloudSim toolkit through the numerical experiments does not take into its consideration the pre-knowledge about the requested VMs (general requesting).

This chapter interests on the comparison with one-dimensional bin-packing algorithms, so that three of the four algorithms have been implemented based on one-dimensional verification (checking) style; which is the number of PEs inside the host just only.

Finally, the following sections introduce the four compared algorithms design, Time-Shared scenarios, Space-Shared scenarios and the discussion of achieving results.

6.2 The Algorithms

This section introduces the four algorithms which are compared with Agent-based algorithm through the CloudSim toolkit; where four Java classes have been implemented based on CloudSim toolkit configuration/design as VM allocation policies to represent the compared algorithms, so that all classes are extended from the Java abstract class called *Vm_Allocation_Policy* [74]. The algorithm as following.

Random algorithm [7, 9]: this algorithm picks one of datacenter hosts based on random fashion without any checking or verification to the resources that host, then it tries to create the requested VM inside the selected host. Algorithm 6.1 presents the algorithmic steps of *Random VM Allocation Policy* in CloudSim toolkit.

Note that, Random policy tries to find a proper host to allocate the Requested VM by picking a random host then try to create the VM in that picked host; if the creation attempt succeeds the policy returns true results, else it picks another host and tries again. The previous steps repeat until the VM creation process happened or the repetition reaches to the triple of host's range to give the Random policy the chance to cover all hosts inside the datacenter, c.f. Algorithm 6.1.

First-Fit algorithm: which picks the first host can fulfill the requirements (specifications) of requested VM. The picking decision (allocation decision) depends on the algorithm's verification criteria; where the criteria through this chapter is a One-Dimensional criterion

this means the allocation algorithm interests on one VM specification (requirement) just only, which is the number of PEs c.f. Section 3.4, the algorithmic steps are introduced through Algorithm 6.2 which is based on the algorithm and definition of First-Fit in [13].

Algorithm 6. 1. Random VM Allocation Policy.

Algorithm 6.1 *Random VM allocation policy*

Variables*: VM_Table Map, Boolean Checked_List[]*
Input*: VM Object*
Output*: Boolean* ***Result***

```
1   Result=False, Checked_List=False, Tries =0,
    range=Host_List.size()
2   If(VM is not already created)
3     Do:
4       For(3 times of (range))
5         idx = Generate a random host Id;
6         If(!Checked_List[idx])
7           Get the Host. Id = idx;
8           Result = Create_VM(Host);
9           Tries++;
10          Break;
11        End If
12      End For
13      If (Result)
14        Update: VM_Table (VM);
15        Result = true
16        Break
17      Else
18        Checked_List[idx] = true;
19      End If
20    While(!Result && Tries< Host_List.size());
21  End If
23  Return Result
```

The First-Fit algorithm in this chapter, depends on one-dimensional verification criteria to find the proper host according to its point of view. Thus, it picks (allocates) the first host has an available PEs equivalent to the number of PEs of requested VM, then try to create a requested VM in current host. If the creation attempt is failed, it avoids the current host from the picking range, then repeats the allocation process phases (picking and creation) again, c.f. Algorithm 6.2.

Algorithm 6. 2. First-Fit 1D VM Allocation Policy.

Algorithm 6.2 *First-Fit 1D VM allocation policy*

Variables*: VM_Table Map, Used_PEs List, Free_PEs List*
Input*: **VM** Object*
Output*: Boolean **Result***

```
1   Result=False/ required_Pes = VM.Pes/ Tries=0/ idx=-1
2   If(VM is not already created)
3       Do:
4           For All (host in Host_List)
5               If(Host.Pes >= required_Pes)
6                   idx= index of the host;
7                   Break
8               End If
9           End For
10          Host = Host_List.get_Host(idx);
11          Result = Create_VM(Host);
12          Tries++;
13          If (Result)
14              Update:
15                  VM_Table (VM)
16                  Used_PEs (required_Pes)
17                  Free_PEs (required_Pes)
18              Result = true
19              Break
20          Else
21              set the Host.Pes in minimum value;
22          End If
23      While (! Result && Tries < Free_PEs. Size);
24  End If
25  Return Result
```

Worst-Fit (Max-Rest) algorithm*:* it is even known as a Max-Rest algorithm; where it picks the host who has a maximum available rest of the resources, this means it allocates the host with worst fit according to the VM specifications. The implementation of the algorithm within the CloudSim toolkit takes into his considerations this chapter specifications, so that the number of PEs of any picked/selected host at least equivalent to the number of PEs of requested VM.

Algorithm 6. 3. Worst-Fit 1D VM Allocation Policy.

Algorithm 6.3 *Worst-Fit 1D VM allocation policy*

Variables*: VM_Table Map, Used_PEs List, Free_PEs List*

Input*: **VM** Object*

Output*: Boolean **Result***

```
1   Result=False/ required_Pes = VM.Pes/ Tries=0/ idx=-1/ max_Free;
2   If(VM is not already created)
3     Do:
4       For All (host in Host_List)
5         If(Host.Pes > max_Free)
6           max_Free = Host.Pes;
7           idx= index of the host;
8         End If
9       End For
10      If (No any available Host)
11        Break;
12      End If
13      Host = Host_List.get_Host(idx);
14      If (max_Free >= required_Pes)
15        Result = Create_VM(Host);
16        Tries++;
17      End If
18      If (Result)
19        Update:
20          VM_Table (VM)
21          Used_PEs (required_Pes)
22          Free_PEs (required_Pes)
23        Result = true
24        Break
25      Else
26        set the Host.Pes in minimum value;
27      End If
28    While (! Result && Tries < Free_PEs. Size);
29  End If
30  Return Result
```

Algorithm 6. 4. Best-Fit 1D VM Allocation Policy.

Algorithm 6.4 *Best-Fit 1D VM allocation policy*

Variables*: VM_Table Map, Used_PEs List, Free_PEs List*
Input*: **VM** Object*
Output*: Boolean **Result***

```
1   Result=False/ required_Pes = VM.Pes/ Tries=0/ idx=-1/ ratio=0
2   If(VM is not already created)
3     Do:
4       For All (host in Host_List)
5         If(Host.Pes >= required_Pes)
6           Temp_ratio = Calculate remaining capacity of the Host PEs
            after add the VM's PEs
7           If (Temp_ratio >= ratio)
8             idx= index of the host;
9             ratio = Temp_ratio;
10          End If
11        End If
12      End For
13      If (No any Host available)
14        Break
15      End If
16      Host = Host_List.get_Host(idx);
17      Result = Create_VM(Host);
18      Tries++;
19      If (Result)
20        Update:
21          VM_Table (VM)
22          Used_PEs (required_Pes)
23          Free_PEs (required_Pes)
24        Result = true
25        Break
26      Else
27        set the Host.Pes in minimum value;
28      End If
29    While (! Result && Tries < Free_PEs. Size);
30  End If
31  Return Result
```

Algorithm 6.3 introduces the pseudo code to the Worst-Fit algorithm according to this chapter specifications, note that these algorithmic steps are modified version of the Worst-Fit algorithm in [13] to be adaptable to the CloudSim toolkit configuration environment and one-dimensional criteria.

Best-Fit algorithm: it tries to pick/allocate the host has the Best-Fit according to requested VM specification; thus it picks a host with the highest utilization rate (the percentage of used resources) and it can fulfill the requested VM requirements. Furthermore, the usage percentage or utilization rate of the host is calculated after adding the requested requirements of VM to the current used/utilized resources within the host [13].

Finally, through this chapter, the utilization rate (usage percentage) is calculated to the number of PEs of the host after adding the number of PEs of requested VM c.f. Algorithm 6.4. again the algorithmic steps of Algorithm 6.4 are based on the Best-Fit definition and algorithm in [13].

Note that, the complexity of the three Bin-Packing algorithms is a *Quadratic Complexity* of order $\boldsymbol{O(n^2)}$ [13], but Agent-based complexity is a *Linear Complexity* of order $\boldsymbol{O(n)}$ c.f. Section 4.5 [19].

The next section presents the comparison between Agent-based policy and these algorithms amongst the Time-Shared scheduler scenarios.

6.3 Time-Shared Scenarios

Three scenarios are introduced through this section based on the Time-Shared scheduler cross host scheduling level amongst the VMs. Time-Shared has dynamic fashion for VMs scheduling c.f. Section 3.6 and allows the sharing of computational cores between the VMs; thus the number of created VMs by using it more than another scheduler (Space-Shared).

The three scenarios depend on fixed general assumptions and condition which presented through Section 5.1 such as One-Datacenter and One-Broker… etc. Also, there are varying conditions and assumptions to each scenario. Furthermore, the Allocation Time and Turnaround Time follow the same definition in Section 5.1.

6.3.1 Scenario I: Hosts vs. VMs Requests

This scenario aims to verify the algorithms' efficiency under the maximum boundary of VMs requesting (over-requesting case). Further, verifies the impact of both allocation time and created VMs on the turnaround time (throughput of the system).

This scenario's conditions and assumptions: First, Using one type of VM. Second, each VM needs: one PE, 1,800 MIPS, 1 GB of RAM and 2000 MHz for the Bandwidth. Third, using two types of Hosts; which are divided equally (50% for each type). Forth, the number of hosts and requested VMs is variety. Fifth, the requested VMs 10 times of hosts number every time.

According to the scenario's figures; the Agent-based algorithm is a superior of all algorithms in the allocation and turnaround times through all practical experiments of this scenario, in contrast, the Random algorithm is an inferior in the terms of time because it depends just on the random picking without any testing criteria. Furthermore, First-Fit, Worst-Fit (Max-rest) and Best-Fit score same Allocation and Turnaround times at maximum boundaries of VMs requesting; because they have similarity in the algorithmic structure and verification criteria (one-dimensional) c.f. figures 6.1 and 6.2.

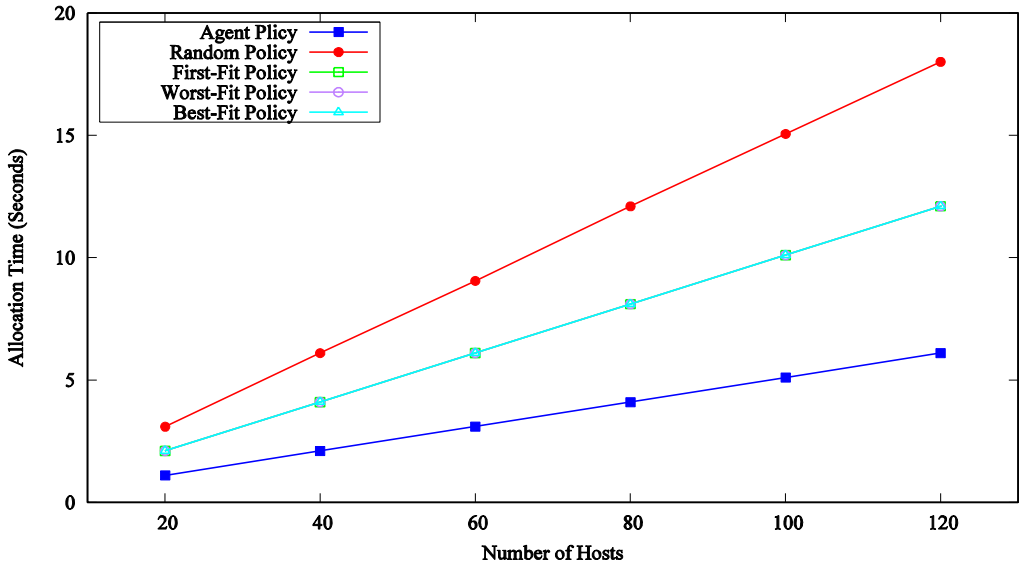

Figure 6. 1: The Allocation Time vs. Number of Hosts.

Due to the similarity of Bin-Packing algorithms (first, worst, best), they have created the same number of VMs in the allocation process among the practical experiments; which leads the

three algorithms to score the same turnaround time especially. Hence, the definition of turnaround time in Section 5.1 depends on the allocation time and execution time of Cloudlets by VMs, additional the execution time (summation) based on three aspects: the number of Cloudlets, number of VMs and type of VMs. This scenario uses just one VM type and one Cloudlet type; which means the execution time for each Cloudlet is equal. Thus, same created VMs with same Cloudlets gives same execution time, consequently, the turnaround time for three algorithms is equal because the allocation time and execution time are equal.

Again, as mentioned through Chapter 5, using a multi-agent system in Agent-based algorithm to build a good awareness about whole datacenter resources (hosts) and the full scan for host resources (VM specifications) by Host-Agent; gives the Agent-based algorithm this superior in time terms because it covers all datacenter hosts (horizontal scan) and all host resources/specifications (vertical scan).

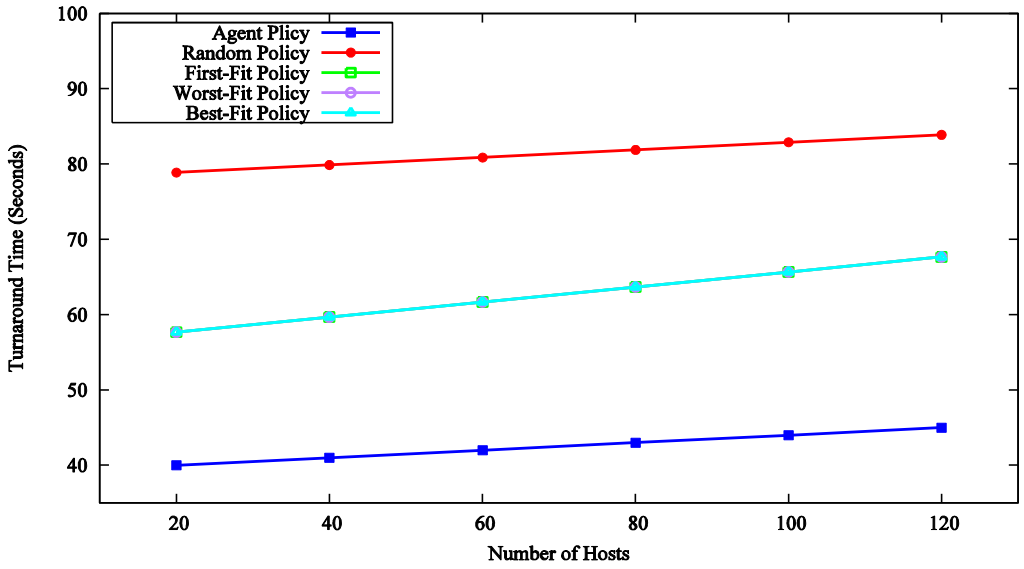

Figure 6. 2: The Turnaround Time of Algorithms.

Finally, the created VMs by all algorithms is divided into two groups: first; Agent-based and random algorithms, second; the Bin-Packing algorithms (First, Worst and Best - Fit). Where the first group created 30% of VMs more than the second one through this scenario, that makes the execution time of first group shorter (better) than the second one which helps Agent-based algorithm to score the minimum turnaround time overall algorithms.

6.3.2 Scenario II: VMs Variety

This scenario is devoted to verifying the impact of VMs variety (more than one type) on the efficiency of all algorithms in terms of time (allocation, turnaround) and allocation VMs (placement, distribution), where it uses three VM types.

Table 6. 1. The VMs Features.

Feature	*VM 1*	*VM 2*	*VM 3*
Image size	10000 MB	20000 MB	30000 MB
RAM	1 GB	2 GB	3 GB
MIPS	1,100	1,600	2,000
Bandwidth	1000	2000	3000
Number of Pes	1	2	3
VMM	XEN	XEN	XEN

The scenario assumptions and conditions: First, using three types of VMs as shown in table 6.1. Second, using three types of Hosts as shown in table 6.2, Third, the number of hosts is fixed (120 hosts); divided into three groups equally, 40 hosts for each type. Forth, the number of requested VMs is varied, but the requesting amount of each VM type is equal.

Table 6. 2. The Hosts Features.

Feature	*Host 1*	*Host 2*	*Host 3*
Number of Pes	3	5	4
MIPS	1000	1500	3000
RAM	10 GB	12 GB	12 GB
Bandwidth	10,000 MHz	10,000 MHz	10,000 MHz
Storage	1 Terabyte	1 Terabyte	1 Terabyte

In allocation time, Agent-based algorithm shows high-performance efficiency with VMs variety compared with other algorithms during all requesting cases (under-request, over-request), also the performance is smoothly and gradually from under-request cases (100 VMs) to over-request cases (900 VMs) c.f. Figure 6.3. Moreover, Agent-based exhibits high stability during the scenario experiments under the impact of VMs' variety, which is opposite of other

algorithms. The random algorithm started close to Agent-based and better than the Bin-Packing algorithms, then made a big jump which makes Bin-Packing algorithms' performance better than it until the end.

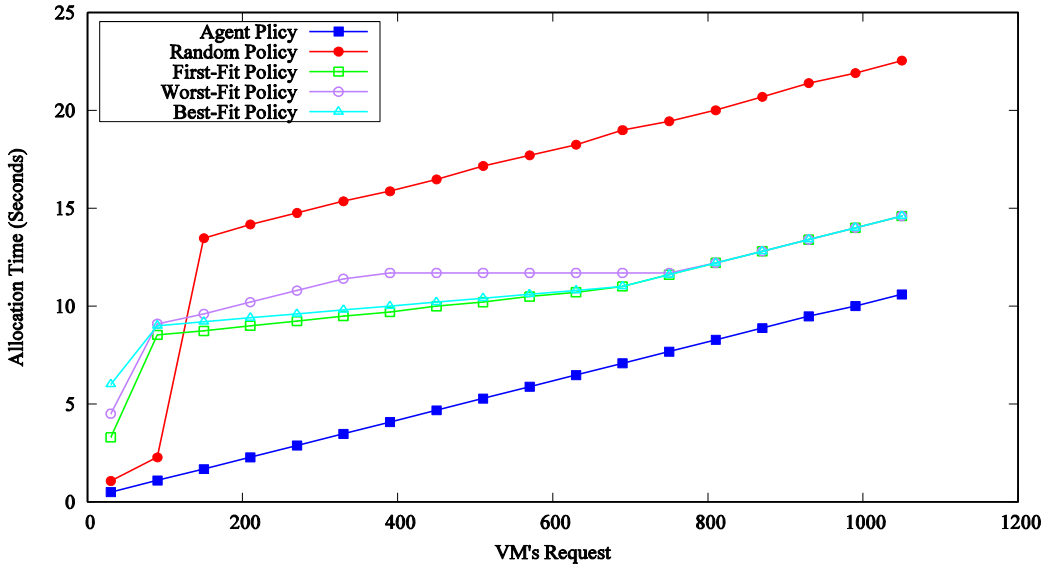

Figure 6. 3: The Allocation Time of Algorithms with VMs Variety.

Due to the similarity of structure and verification criteria between the Bin-Packing (First, Worst, Best), they achieved the same allocation time in over-request cases (Max-boundaries) like Sub-Section 6.3.1. But, different allocation times are achieved by Bin-Packing algorithms in under-request cases, this means the VMs variety has a clear impact on Bin-Packing algorithms especially in under-request cases c.f. figure 6.3.

In turnaround time, Agent-based remains achieving best times over all requesting cases (under, over) compared with other algorithms, especially in over-request cases such as 1000 requested VMs; due to the difference between all algorithms and Agent-based in allocation time and rate of created VMs c.f. figures 6.4-6.9. Noticeably, the gap between random algorithm and Agent-based almost stable or fixed after 150 requested VMs because almost there is no difference between them in the created VMs, also no algorithm could create more than Agent-based in any VM types unless the random algorithm which created VMs in the second type (VM 2) c.f. figures 6.5 and 6.6.

The stairs shape of all algorithm curves (lines) in turnaround time figure comes from the irregular distribution of Cloudlet over the created VMs. In the regular distribution, the created VMs equal the requested VMs (optimal distribution), which means the turnaround line

graded smoothly corresponding to the requested VMs. Thus, the steps or leaps appear (show up) when the gap between the created VMs and requested VMs increase (non-optimal distribution) in some requesting cases c.f. figure 6.4.

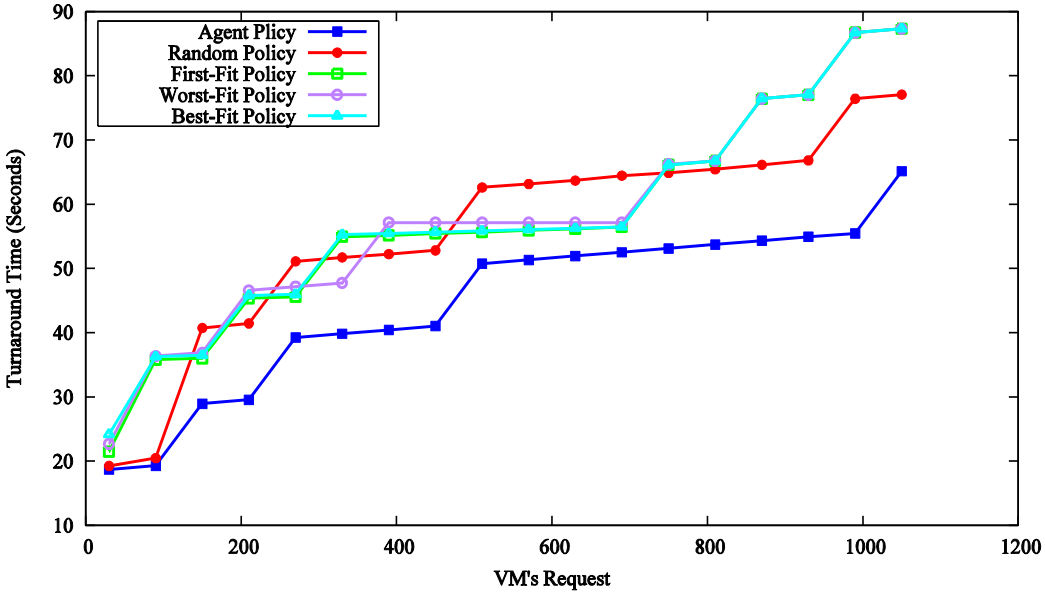

Figure 6. 4: The Turnaround Time of Algorithms with VMs Variety.

The figures 6.5-6.9 show the distribution of created VMs amongst the three types and the VMs creation rate (which refers to the total number of created VMs during the numerical experiments or inter the scenario) for all algorithms.

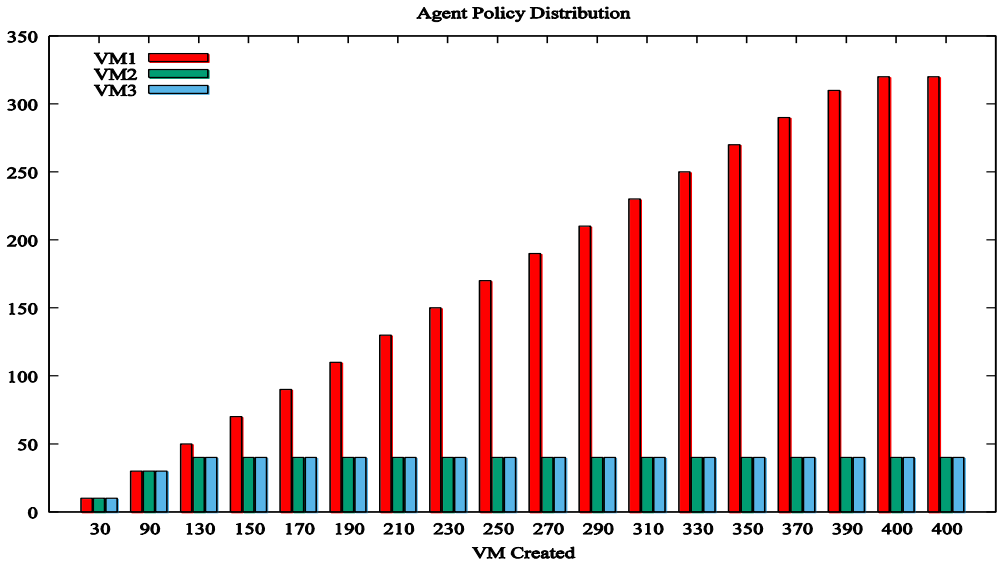

Figure 6. 5: The Distribution of Agent-based policy to the Created VMs.

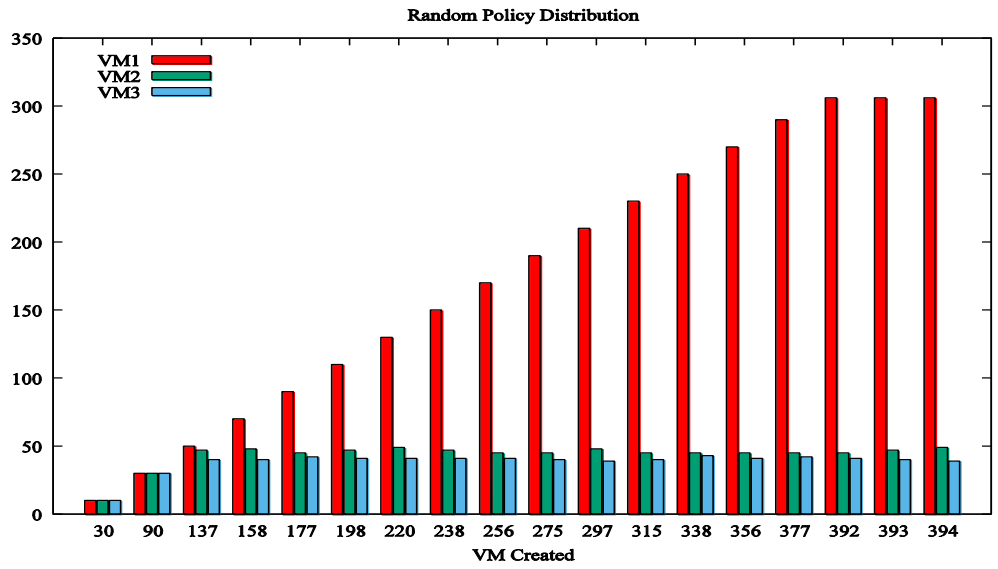

Figure 6. 6: The Distribution of Random Algorithm to the Created VMs.

The Bin-Packing algorithms achieved almost same turnaround times and same line shape especially in over-request cases, due to the allocation times were very close under-request cases or same in over-request cases and the created VMs almost same for all. However, the turnaround line is not stable and not smoothly (many jumps) more than Agent-based and random algorithms, because of the creation rate of Bin-Packing algorithms less than Agent-based and random.

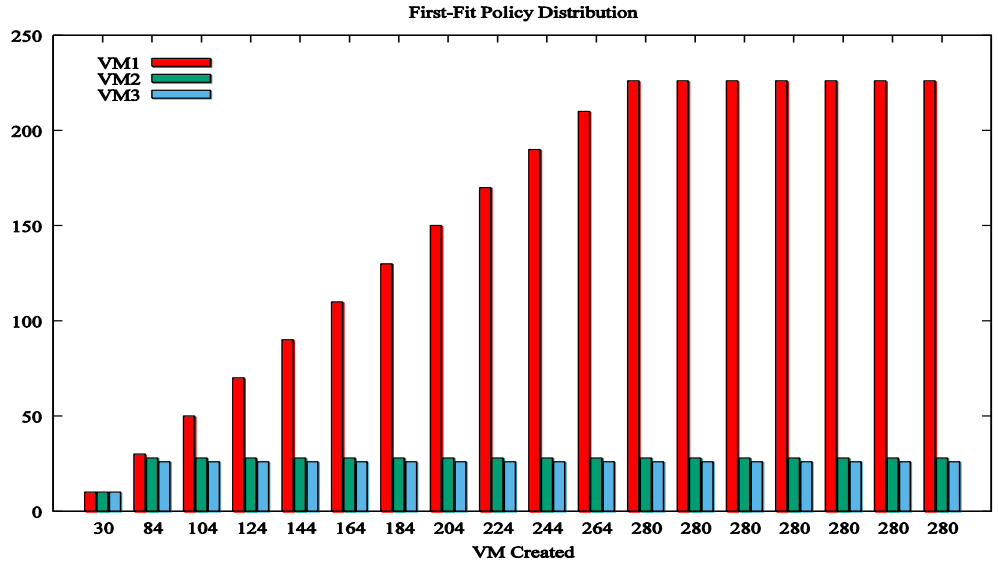

Figure 6. 7: The Distribution of the First-Fit Algorithm to the Created VMs.

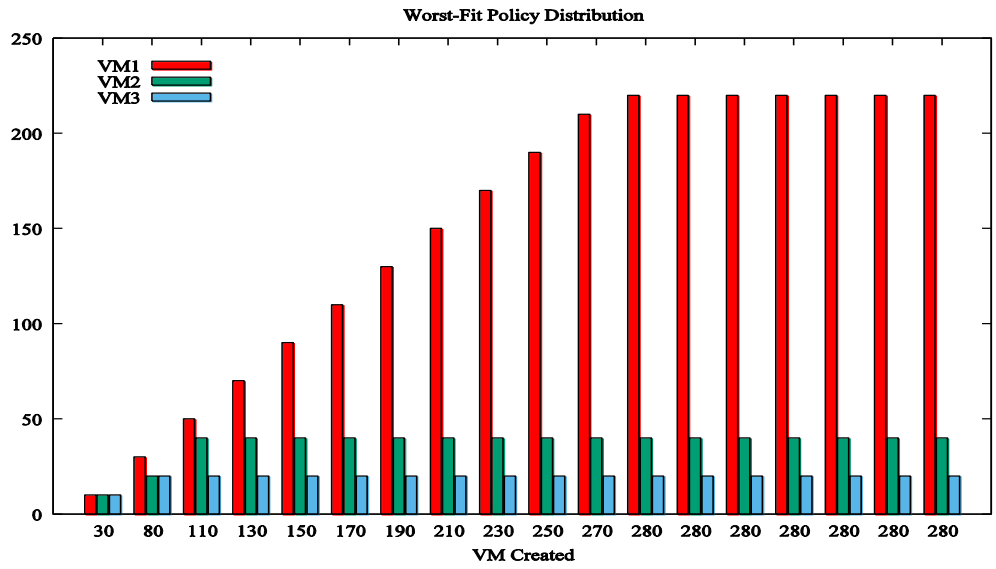

Figure 6. 8: The Distribution of Worst-Fit (Max-Rest) Algorithm to the Created VMs.

According to the Distribution figures (6.5 – 6.9), there is a big gap in VMs creation rate between: firstly, Agent-based and random algorithms which created around 400 VMs at the end. And secondly, the Bin-Packing algorithms that created around 280 VMs. Agent-based has achieved the highest creation rate, and the distribution among the VMs types, in general, is well-balanced compared with other algorithms especially over second and third types.

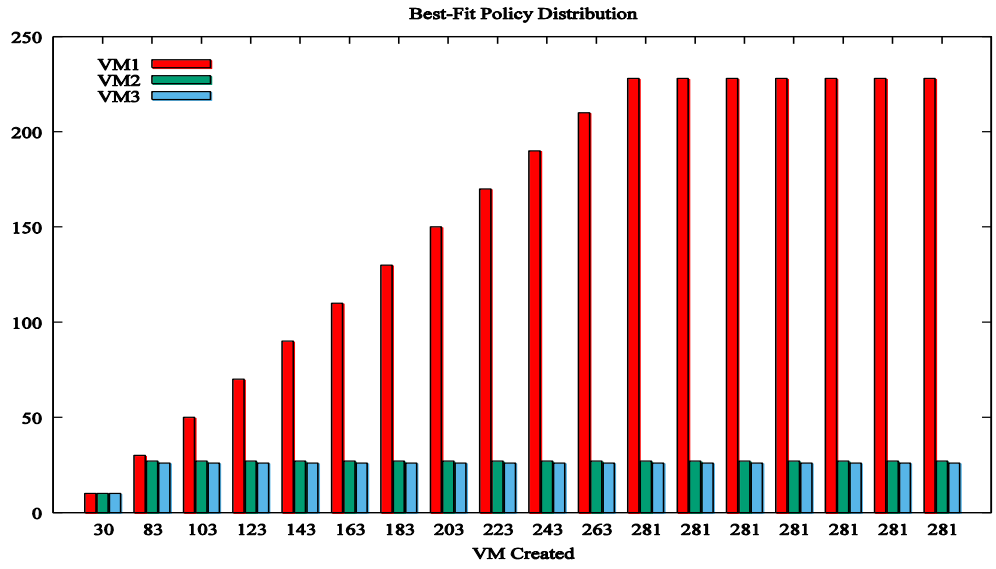

Figure 6. 9: The Distribution of Best-Fit Algorithm to the Created VMs.

Finally, the Agent-based policy has exhibited high efficiency corresponds to other algorithms during this scenario in terms of: (i) time including allocation and turnaround times c.f. figures 6.3 and 6.4, and (ii) allocation process including the VMs creation rate and VMs types distribution c.f. figures 6.5-6.9.

6.3.3 Scenario III: VMs Variety vs. Host Variety

This scenario aims to imitate the real-life situation, where the real cloud datacenter has a variety of hosts' types and the VMs are varying through the requesting process. In terms of that, this scenario has five host types in the datacenter and five VM types are requested through the requesting cases in numerical experiments.

Table 6. 3. The Five VMs Features.

Feature	*VM 1*	*VM 2*	*VM 3*	*VM 4*	*VM 5*
Image size	10000 MB	10000 MB	10000 MB	10000 MB	10000 MB
RAM	1 GB	2 GB	3 GB	3 GB	3 GB
MIPS	900	1,800	1,300	1,400	1,000
Bandwidth	1000	2000	2000	3000	3000
Number of Pes	1	1	2	3	4
VMM	XEN	XEN	XEN	XEN	XEN

Conditions and assumptions of the scenario: First, using five types of VMs as shown in table 6.3. Second, using five types of Hosts as shown in table 6.4, Third, the number of hosts is fixed (200 hosts); divided into five groups equally, 40 hosts for each type. Forth, the number of requested VMs is varied, but the requesting amount of each VM type is equal.

Table 6. 4. The Five Hosts Features.

Feature	*Host 1*	*Host 2*	*Host 3*	*Host 4*	*Host 5*
Number of Pes	3	5	4	3	2
MIPS	1000	1000	1500	2000	3000
RAM	4 GB	6 GB	10 GB	10 GB	12 GB
Bandwidth	10,000 MHz	10,000 MHz	10,000 MHz	10,000 MHz	10,000 MHz
Storage	1 Terabyte	1 Terabyte	1 Terabyte	1 Terabyte	1 Terabyte

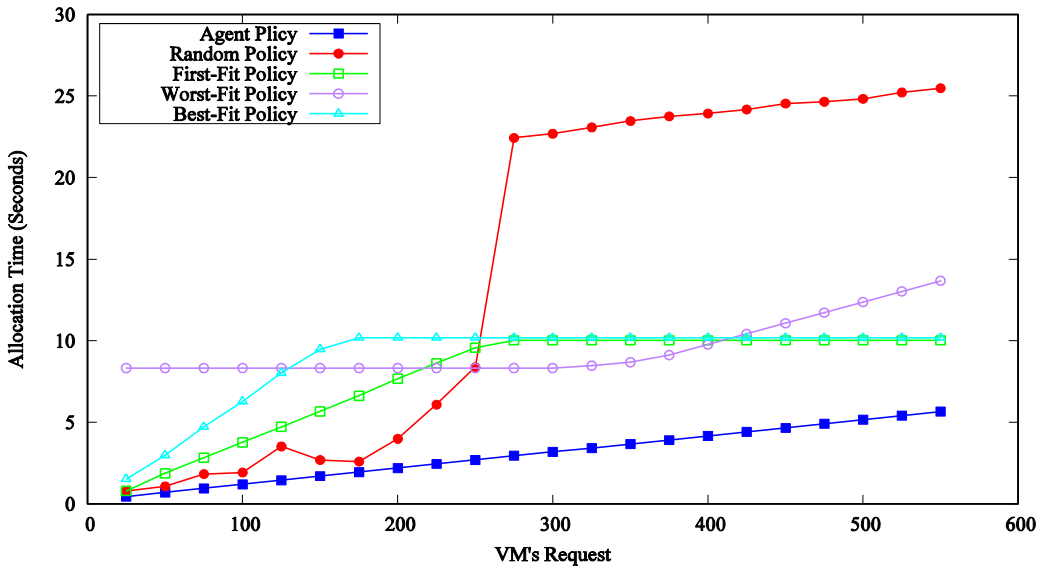

Figure 6. 10: The Allocation Time for Algorithms.

In figure 6.10, the impact of Hosts and VMs diversity is very obvious over the allocation time (allocation process) especially on Bin-Packing algorithms, where the differences between all Bin-Packing algorithms appear exactly during the under-request cases. The Worst-fit during all under-request cases shows stable functionality and scored almost same values of allocation time, but the values were higher than others Bin-Packing algorithms, especially in beginning, then the values of allocation time starting rise up (350 VMs) gradually through over-request cases until the end.

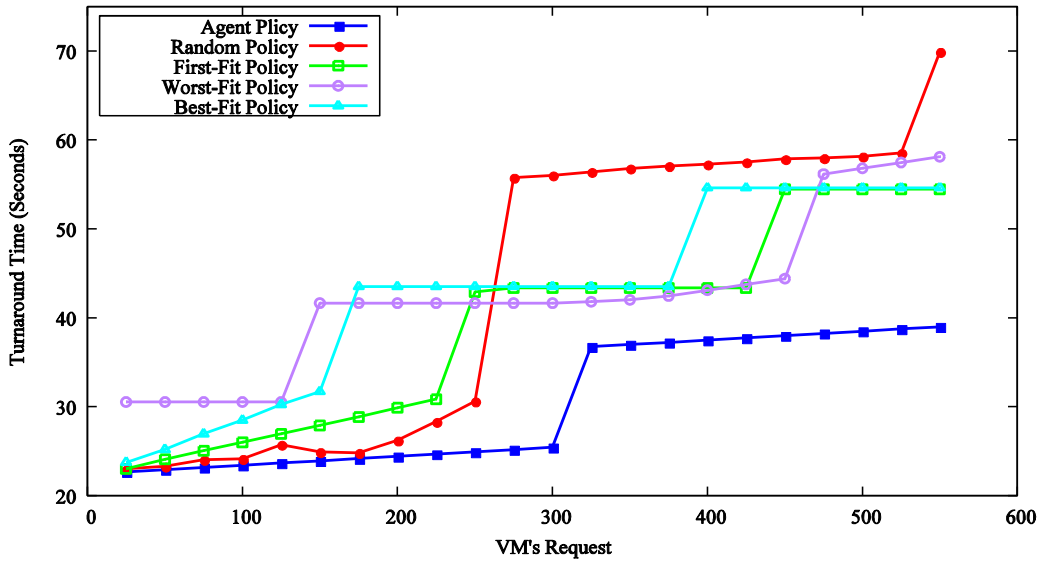

Figure 6. 11: The Turnaround Time for Algorithms.

The First-Fit and Best-Fit algorithms in figure 6.10, score very close values to the Agent-based and Random algorithms, in the beginning, then the values of allocation time starting growth up especially for the Best-Fit during under-request cases of VMs. Both of them in over-request cases (after 275 VMs) show very stable and steady performance, where almost each of them scores the same values of allocation time through over-request cases. Overall Bin-Packing algorithms, the First-Fit showed the best performance over all request cases during this scenario.

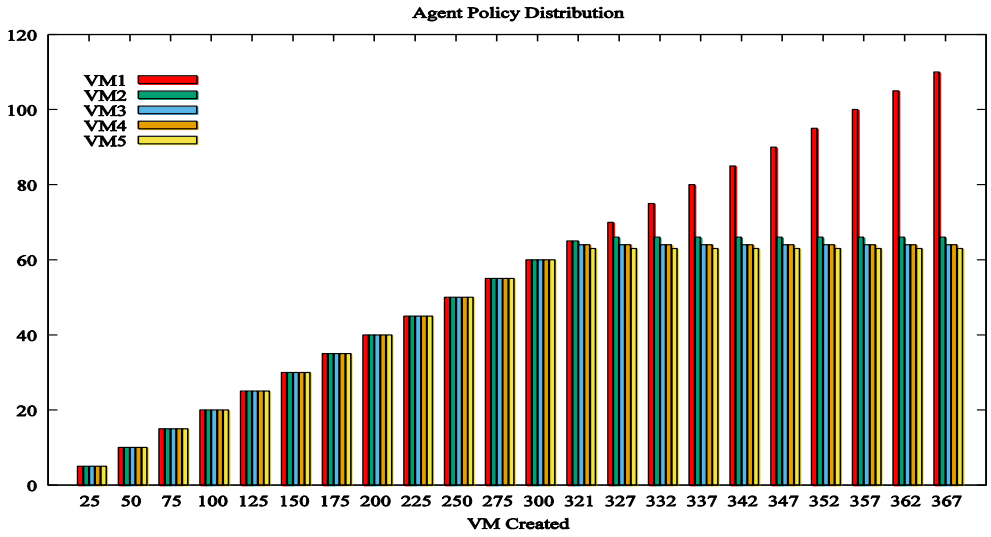

Figure 6. 12: The Distribution of Agent-based Algorithm to the Created VMs.

The Random algorithm values in allocation time through under-request cases (25 – 250 VMs) were better/lower than all Bin-Packing algorithms, but after that, a big jump happened which makes the Random algorithm values in over-request cases worse/ higher than all Bin-Packing algorithms. Agent-based again shows high efficiency in allocation time over all requesting cases, furthermore, it exhibits very steady and stable performance by the smoothly graded values of allocation time during the numerical experiments of this scenario c.f. figure 6.10.

Turnaround time depends on allocation time, cloudlets and created VMs at least c.f. Section 5.1. Hence, we can avoid the impact of cloudlets because all scenarios use just one type of cloudlets; due to that the allocation time and created VMs have the main impact over turnaround time.

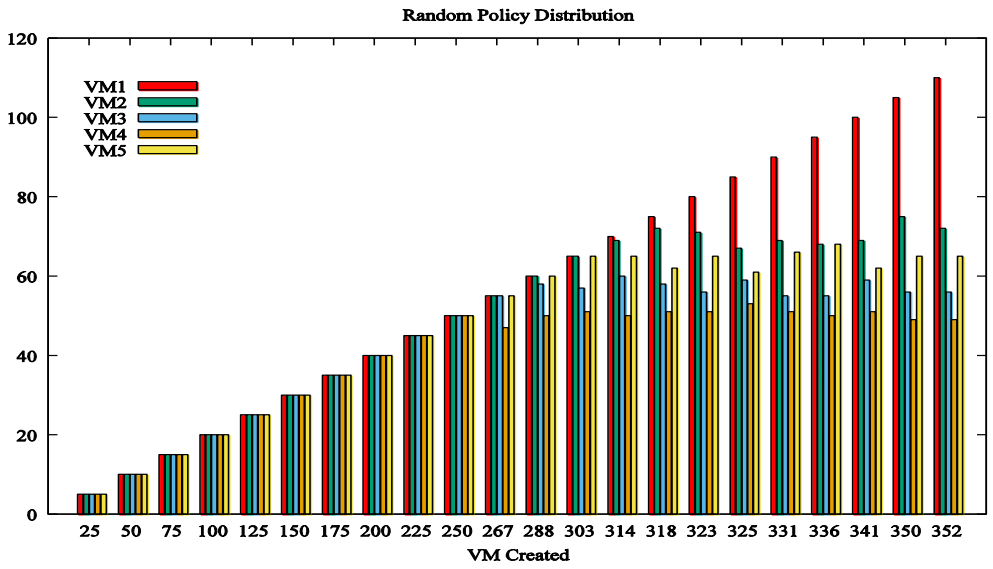

Figure 6. 13: The Distribution of Random Algorithm to the Created VMs.

Figure 6.11, the Agent-based algorithm shows high efficiency in the turnaround time through all the requesting cases, further it scores the best values of turnaround time during all numerical experiments of the scenario compare with other algorithms. The Agent-based has achieved the best values of allocation time c.f. figure 6.10 and it created the maximum number of VMs with a balanced distribution of VMs' type's c.f. figure 6.12, which leads Agent-based to achieve the best values of turnaround time.

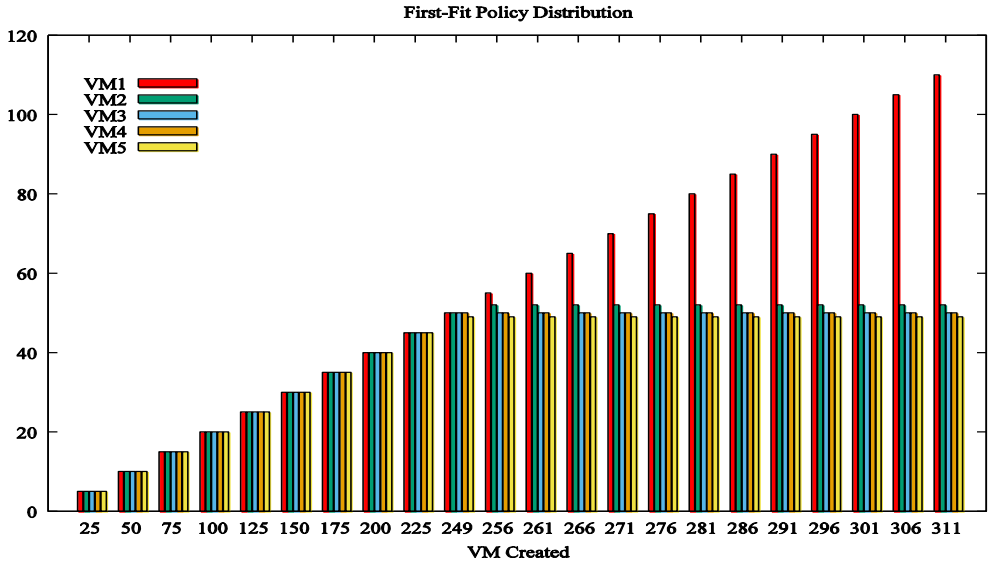

Figure 6. 14: The Distribution of the First-Fit Algorithm to the Created VMs.

A good allocation time for Random algorithm through under-request cases c.f. figure 6.10 with a good VMs creation rate, where Random algorithm takes the second place in VM creation process c.f. figure 6.13; this helps it to achieve a turnaround time values better than all Bin-Packing algorithms. but the bad allocation time (after 250 VMs) of Random algorithm obviously impacts on achieving turnaround time during over-request cases, which makes the turnaround time of all Bin-packing algorithms better than Random one.

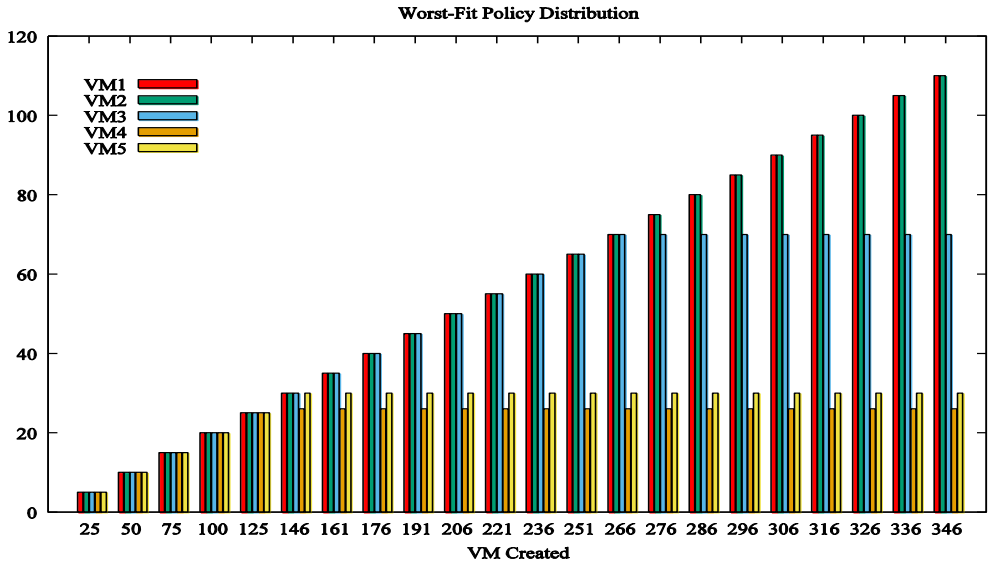

Figure 6. 15: The Distribution of Worst-Fit Algorithm to the Created VMs.

The Bin-Packing algorithms in turnaround time figure 6.11 take the same manner in allocation time figure 6.10, where the First-Fit and Best-Fit algorithms take the leading in the beginning over the Worst-Fit one and at the end, they return to the leading again. But in some requesting cases in the middle such as 275, 300, 325 and 350 VMs the Worst-Fit algorithm takes the leading over the other two algorithms especially over Best-Fit one. Overall Bin-Packing algorithms the First-Fit one shows good performance corresponding to others two in turnaround time over all requesting cases.

The jumps/leaps in figure 6.11 come from irregular (optimal) distribution of Cloudlets over the created VMs as mentioned before. Agent-based has the highest creation rate with a balanced distribution of VMs' types c.f. figure 6.12, this helps Agent-based to make just one jump/leap over whole the turnaround time line which means the lowest jump/leap rate compare with other algorithms.

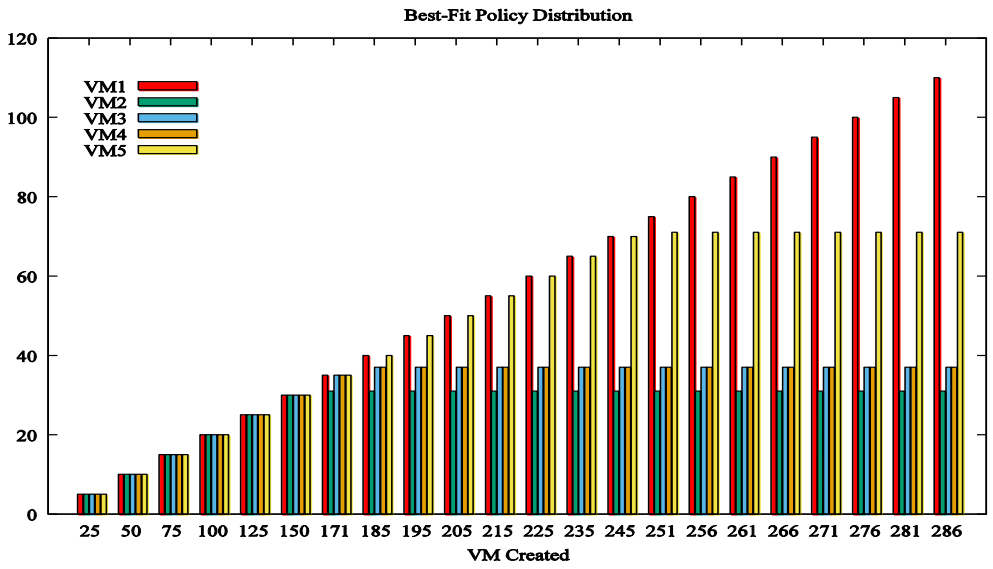

Figure 6. 16: The Distribution of Best-Fit Algorithm to the Created VMs.

According to figures 6.14-6.16, the Worst-Fit algorithm takes the third place of VM creation rate after Agent-based and Random but the distribution of VMs' types is an average balance, and the last place of VM creation rate went to Best-Fit one with average balance distribution of VMs' types. The First-Fit algorithm takes the fourth place and gives smoothly balance the distribution of VMs' types.

The impact of hosts and VMs diversity/variety is very obvious on all algorithms during all numerical experiments of this scenario, moreover the impact of diversity/variety shows up the tiny differences between the Bin-Packing algorithms in terms of time (including allocation and turnaround) and creation process (creation rate, distribution), where the similarity of structure and verification criteria makes that very difficult through previous scenarios.

Finally, the good awareness about all datacenter resources which built by using a multi-agent system gives the Agent-based algorithm (policy) the leading over all measurement aspects including allocation time, turnaround time, VM creation rate and balanced distribution of VMs' types in this scenario. Also, this gives induction about the Agent-based potential to work in a real-life cloud datacenter because it shows high efficiency and functionality with complicated scenarios like this one.

After introducing three scenarios through this section, the next section will introduce another three scenarios for the same purposes of this section scenarios, but with using the Space-Shared scheduler in host level.

6.4 Space-Shared Scenarios

Space-Shared has the static/fixed fashion for VMs scheduling c.f. Section 3.6 which is contrary to the Time-Shared one, further it does not allow the sharing of computational cores between the VMs; thus the number of created VMs by using it less than Time-Shared scheduler. Three scenarios will introduce through this section based on the Space-Shared scheduler cross host scheduling level amongst the VMs.

The three scenarios depend on fixed general assumptions and condition which presented through Section 5.1 such as One-Datacenter and One-Broker… etc. Also, there are varying conditions and assumptions to each scenario. Furthermore, the Allocation Time and Turnaround Time follow the same definition in Section 5.1.

6.4.1 Scenario I: Fixed Hosts

In the scenario, the requesting cases are verified by using a fixed number of hosts in the datacenter, where it aims to verify the functionality of all algorithms over the resources pooling (datacenter) from under-requested cases up to the over-requested cases with using the Space-Shared scheduler.

The conditions and assumptions: First, Using one type of VM. Second, each VM needs: one PE, 1,700 MIPS, 1 GB of RAM and 2000 MHz for the Bandwidth. Third, using two types of Hosts; which are divided equally (50% for each type). Forth, the number of requested VMs is variety. Fifth, the number of hosts (datacenter capacity) is fixed.

The configuration of scenario is not complicated because it just uses one VM type and the hosts have plenty of resources corresponds to the VM's specifications (requirements). According to that, the requesting-cases can be verified in smoothly way and gradually to test the efficiency of all algorithms through these cases.

Agent-based shows high efficiency over all the requesting-cases of VMs start from under-request cases until the over-request especially in the allocation time, also it shows smoothly behavior and stability over all cases in allocation time and turnaround time. Moreover, it achieved the best time in the allocation and turnaround times compared with all other

algorithms, especially through over-request cases such as 170 VMs c.f. figures 6.17 and 6.18; which means Agent-based algorithm technique is the best in the time terms.

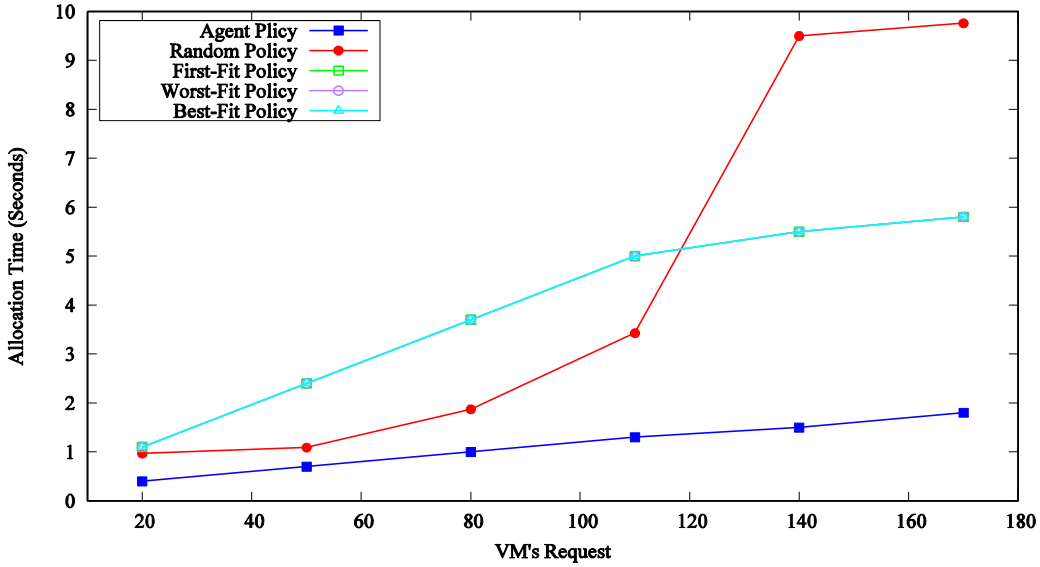

Figure 6. 17: The Allocation Time Corresponding to VM's Request.

The Bin-Packing algorithms (First, Worst, and Best) achieved same time values over all requesting-cases in allocation and turnaround time; because they have similarity in the algorithmic structure (conventional) and verification criteria (one-dimensional). Moreover, they created the same number of VMs through all requesting-cases same as all other algorithms, this means the execution time is the same, which leads to make the turnaround time equivalent to all Bin-Packing algorithms at the end. In addition, the configuration of scenario is simple, so the differences between Bin-Packing algorithms will not appear.

The random algorithm shows good efficiency and smoothly achieving compare with Bin-Packing algorithms through under-request cases in allocation time c.f. figure 6.17, and turnaround time c.f. figure 6.18 such as 80 requested VMs, but it started struggling through over-request cases such as 150 requested VMs, where big gap appeared and the Bin-Packing took the advantage over it especially in allocation time c.f. figures 6.17 and 6.18.

The created VMs through all scenario experiments over all requesting cases for all algorithms were equivalent, also they less than the created VMs in the same scenario if the Time-Shared scheduler is used; because of the nature of Space-Shared scheduler. Moreover, the low VMs creation rate, low requested VMs and low rate of Cloudlets of this scenario compared with

some Time-Shared scenarios (c.f. previous section), prevents the stairs shape to appear (just one leap/jump) in turnaround time figure 6.18.

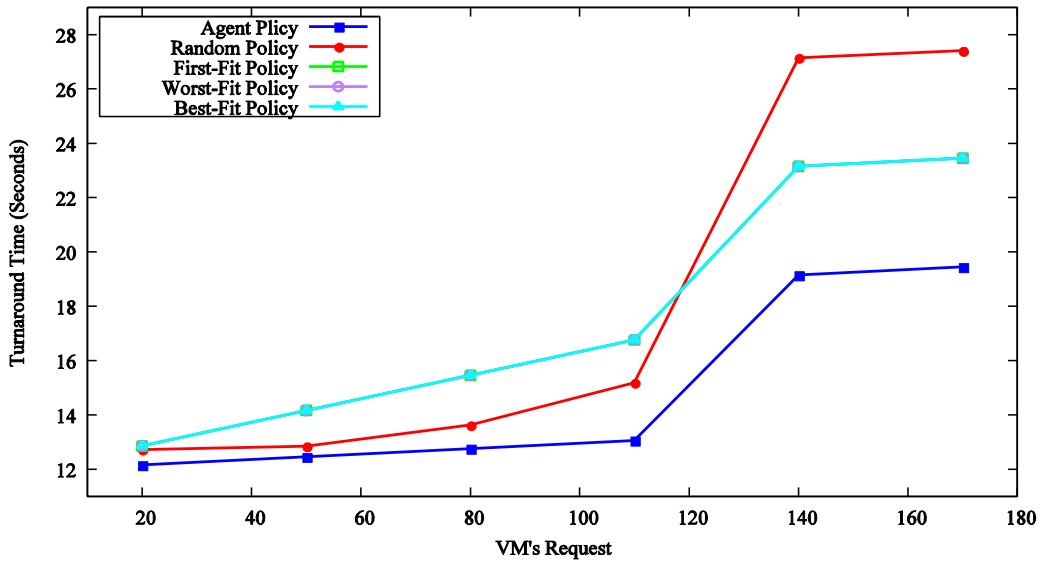

Figure 6. 18: The Turnaround Time Corresponding to VM's Request.

Note that, the disappear of stairs shape (leaps/jumps) through turnaround time in figure 6.18 does not mean the distribution of cloudlets over the created VMs is regular or optimal, but the using of Space-Shared scheduler makes the datacenter reach to the saturated situation early, this mitigates the gaps between requested VMs and created VMs. Which means the rates of VMs requesting and cloudlets will not grow more until the irregular (non-optimal) distribution of cloudlets (over-request cases), consequent no stairs shape appear in turnaround time figure 6.18.

Eventually, Agent-based policy showed the advantages and efficiency of its algorithm through all requesting cases during this scenario in time terms over all other algorithms.

6.4.2 Scenario II: VMs Variety

This scenario is devoted for verifying the impact of VMs variety with Space-Shared on the efficiency of all algorithms in terms of time (including: allocation, turnaround) and allocation VMs (placement, distribution), where it uses three VM types.

Table 6. 5. The VMs Features.

Feature	*VM 1*	*VM 2*	*VM 3*
Image size	10000 MB	20000 MB	30000 MB
RAM	1 GB	2 GB	3 GB
MIPS	1,100	1,600	2,000
Bandwidth	1000	2000	3000
Number of Pes	1	2	3
VMM	XEN	XEN	XEN

Assumptions and conditions of the scenario: First, using three types of VMs as shown in table 6.5. Second, using three types of Hosts as shown in table 6.6, Third, the number of hosts is fixed (120 hosts); divided into three groups equally, 40 hosts for each type. Forth, the number of requested VMs is varied, but the requesting amount of each VM type is equal.

Table 6. 6. The Three Hosts Features.

Feature	*Host 1*	*Host 2*	*Host 3*
Number of Pes	3	5	4
MIPS	1000	1500	3000
RAM	10 GB	12 GB	12 GB
Bandwidth	10,000 MHz	10,000 MHz	10,000 MHz
Storage	1 Terabyte	1 Terabyte	1 Terabyte

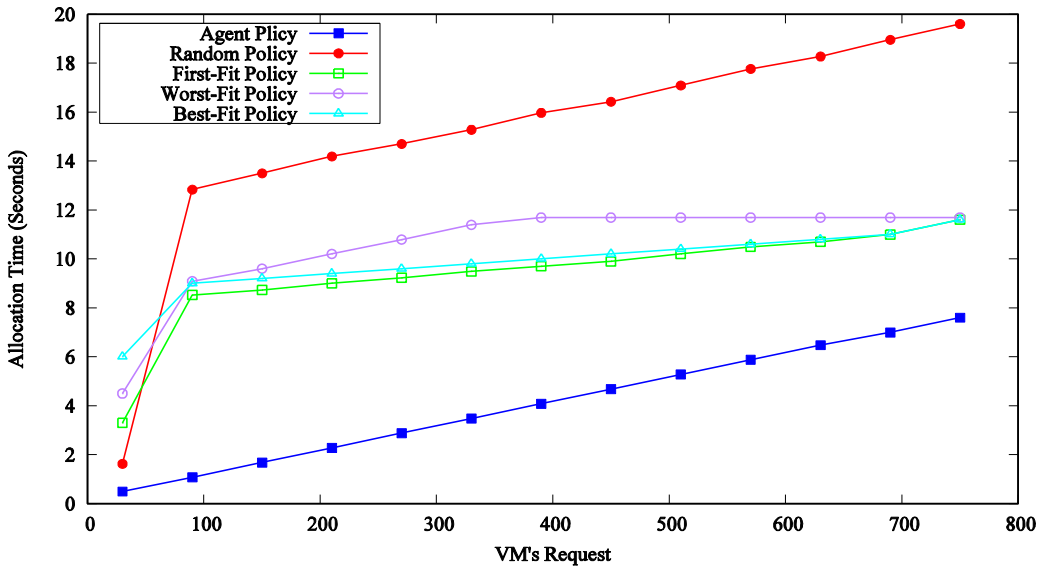

Figure 6. 19: The Allocation Time Corresponding to VMs request.

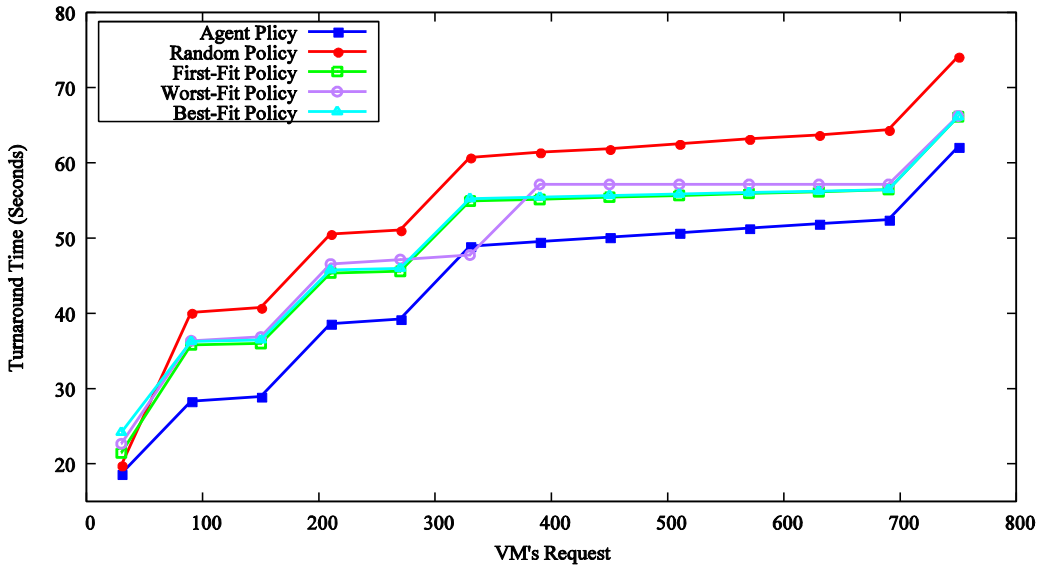

Figure 6. 20: The Turnaround Time Corresponding to VMs request.

The agent-based algorithm shows high-performance efficiency in allocation time with VMs variety compared with other algorithms during all requesting cases (under-request, over-request), also the performance is smooth and gradually from under-request cases (100 VMs) to over-request cases (700 VMs) c.f. figure 6.19.

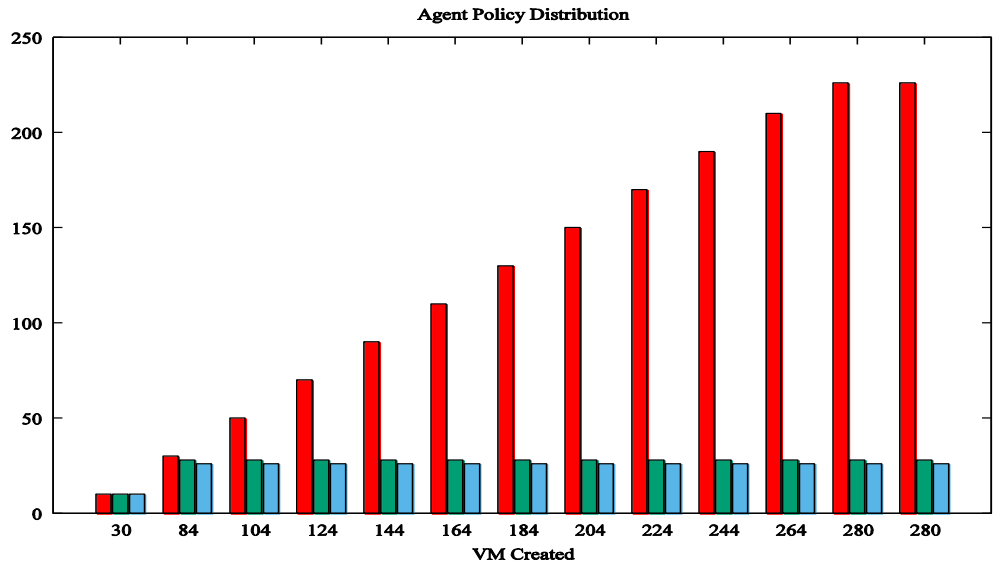

Figure 6. 21: The Distribution of Agent-based Algorithm to the Created VMs.

Moreover, Agent-based exhibits high stability during the scenario experiments under the impact of VMs' variety, which is opposite of other algorithms. The random algorithm started close to Agent-based and better than the Bin-Packing algorithms (30 VMs), then made a big

jump (90 VMs) which makes Bin-Packing algorithms' performance better than it until the end, c.f. figure 6.19.

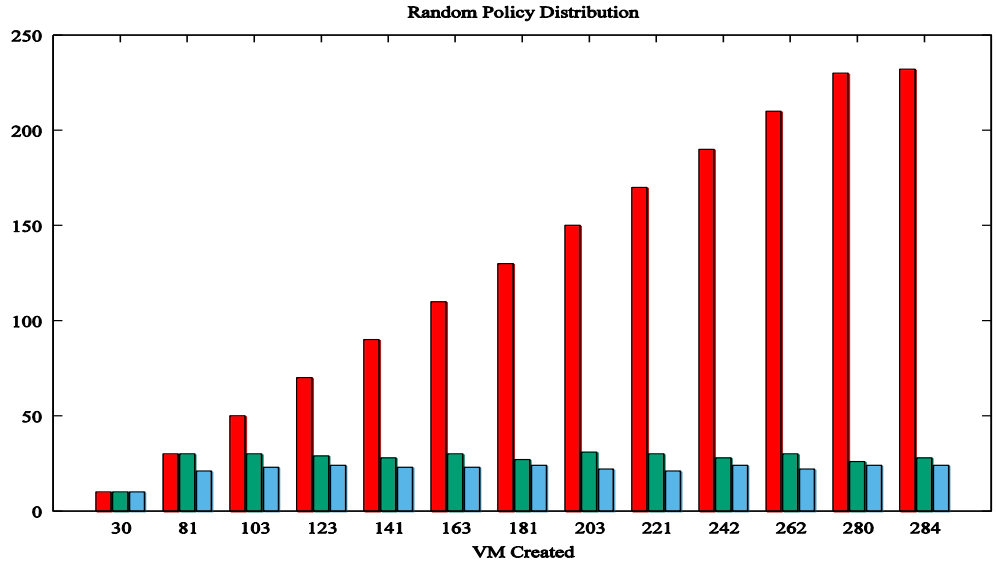

Figure 6. 22: The Distribution of Random Algorithm to the Created VMs.

Despite of the similarity of structure and verification criteria between the Bin-Packing (First, Worst, Best), they achieved different allocation time in under-request cases. But, same allocation time value is achieved by Bin-Packing algorithms at the end, this means the VMs variety has a clear impact on Bin-Packing algorithms especially in under-request cases c.f. figure 6.19.

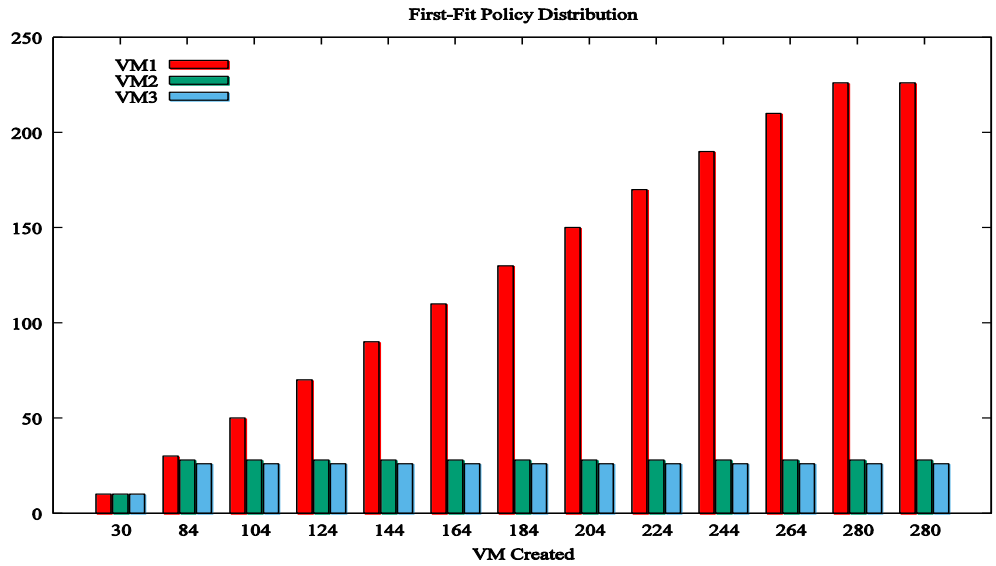

Figure 6. 23: The Distribution of the First-Fit Algorithm to the Created VMs.

Agent-based remains achieving best times over all requesting cases (under, over) in turnaround time c.f. figure 6.20 compares with other algorithms except in one point (330 VMs) where the Worst-Fit takes the leading; due to the difference between all algorithms and Agent-based in allocation time c.f. figure 6.20.

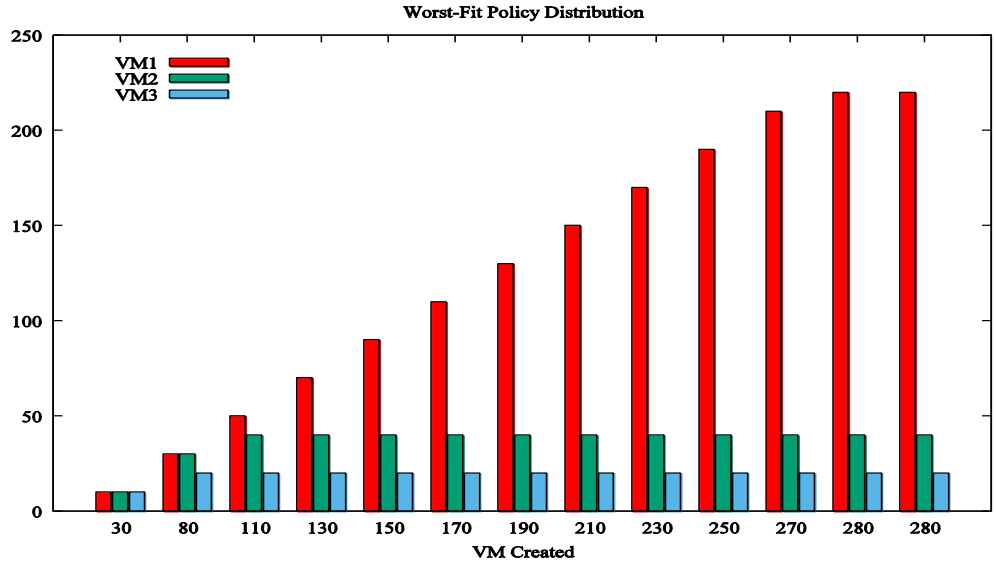

Figure 6. 24: The Distribution of Worst-Fit (Max-rest) Algorithm to the Created VMs.

Noticeably, the impact of using the Space-Shared scheduler is very obvious in the allocation process, because all algorithms create the almost same amount of created VMs and the distribution very close for all of them c.f. figures 6.21-6.25.

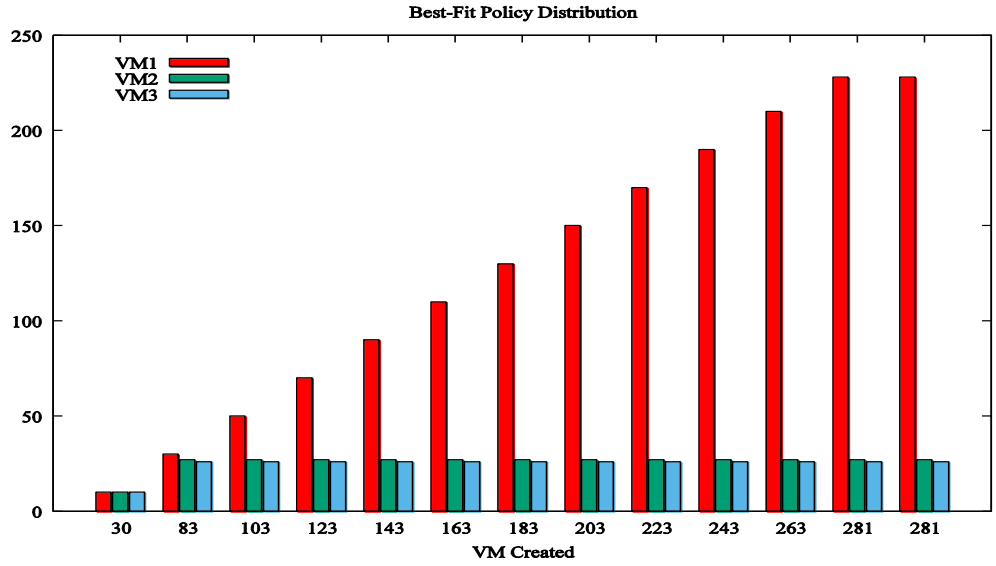

Figure 6. 25: The Distribution of Best-Fit Algorithm to the Created VMs.

The stairs shape of all algorithm curves (lines) in turnaround time figure comes from the irregular distribution of Cloudlet over the created VMs. Where in the regular distribution, the created VMs equal the requested VMs (optimal distribution), which means the turnaround line graded smoothly corresponding to the requested VMs. Thus, the steps or leaps appear (show up) when the gap between the created VMs and requested VMs increase (non-optimal distribution) in some requesting cases c.f. figure 6.20.

The First-Fit and Best-Fit algorithms achieved almost same turnaround times and same line shape especially in over-request cases, due to the allocation times were very close under-request cases and same in over-request cases. But, Worst-Fit achieved different turnaround time and line shape compare with others two algorithms in under-request cases, before getting close to them in over-request cases then joining them at the end.

According to the Distribution figures (6.21 – 6.25), all algorithms were very close in VMs creation rate and distribution of VMs' types, this means the using of Space-Shared avoids the differences between the Bin-Packing algorithms and Agent-based in the same scenario with using the Time-Shared scheduler. Due to the similarity between Agent-based (First-Pass-Fit) and First-Fit in selection criteria makes both algorithms to create exactly the same number of VMs with the same distribution of VMs' types during this scenario c.f. figures 6.21 and 6.23.

Finally, the Agent-based policy has exhibited high efficiency corresponds to other algorithms during this scenario in terms of time, including allocation and turnaround times c.f. figures 6.19 and 6.20. Furthermore, all algorithms have occupied the same amount of resources in RAM and Bandwidth, but Agent-based and First-Fit have occupied more than all others in MIPS (computational power).

6.4.3 Scenario III: VMs Variety vs. Host Variety

The real-life situation is imitated through this scenario, where the real cloud datacenter has a variety of hosts' types and the VMs are varying through the requesting process. In terms of that, this scenario has five host types in the datacenter and five VM types are requested through the requesting cases in numerical experiments, same as Sub-Section 6.3.3.

Table 6. 7. The Five VMs Features.

Feature	*VM 1*	*VM 2*	*VM 3*	*VM 4*	*VM 5*
Image size	10000 MB	10000 MB	10000 MB	10000 MB	10000 MB
RAM	1 GB	2 GB	3 GB	3 GB	3 GB
MIPS	900	1,800	1,300	1,400	1,000
Bandwidth	1000	2000	2000	3000	3000
Number of Pes	1	1	2	3	4
VMM	XEN	XEN	XEN	XEN	XEN

Conditions and assumptions (c.f. Sub-Section 6.3.3): First, using five types of VMs as shown in table 6.7. Second, using five types of Hosts as shown in table 6.8, Third, the number of hosts is fixed (200 hosts); divided into five groups equally, 40 hosts for each type. Forth, the number of requested VMs is varied, but the requesting amount of each VM type is equal.

Table 6. 8. The Five Hosts Features.

Feature	*Host 1*	*Host 2*	*Host 3*	*Host 4*	*Host 5*
Number of Pes	3	5	4	3	2
MIPS	1000	1000	1500	2000	3000
RAM	4 GB	6 GB	10 GB	10 GB	12 GB
Bandwidth	10,000 MHz	10,000 MHz	10,000 MHz	10,000 MHz	10,000 MHz
Storage	1 Terabyte	1 Terabyte	1 Terabyte	1 Terabyte	1 Terabyte

The impact of Hosts and VMs diversity is very obvious over the allocation time (allocation process) especially on Bin-Packing algorithms, where the differences between all Bin-Packing algorithms appear exactly during the under-request cases. The Worst-fit during all under-request cases shows stable performance and scored almost same values of allocation time, but the values were higher than others Bin-Packing algorithms especially in beginning, then the values of allocation time starting rise up (350 VMs) gradually through over-request cases until the end, c.f. figure 6.26.

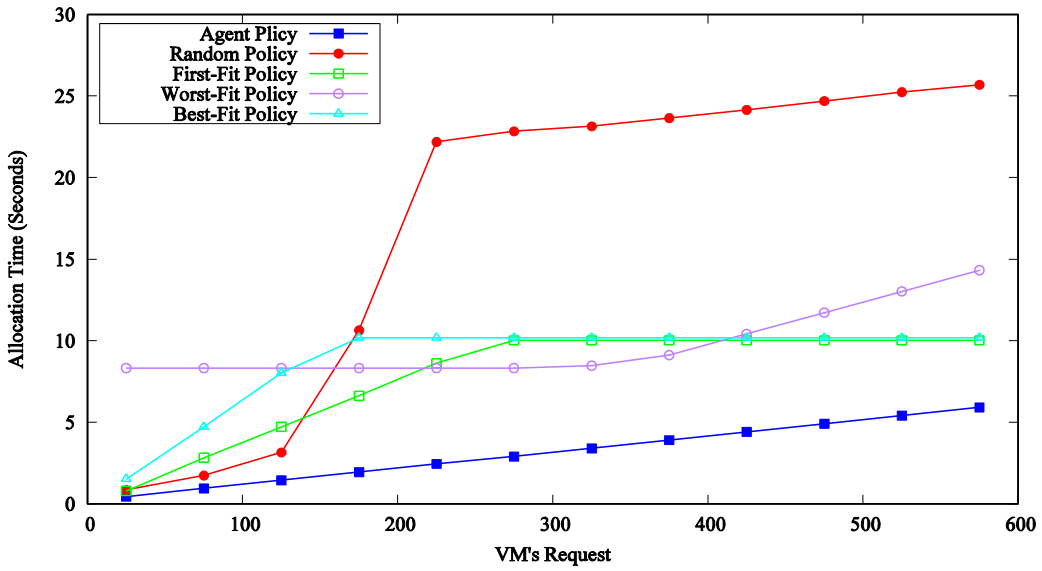

Figure 6. 26: The Allocation Time of Algorithms vs. VMs Request.

According to figure 6.26, the First-Fit and Best-Fit algorithms score very close values to the Agent-based and Random algorithms, in the beginning, then the values of allocation time starting growth up, especially for the Best-Fit during under-request cases of VMs (even higher than Worst-Fit). Both of them in over-request cases (after 275 VMs) show very stable and steady performance, where almost each of them scores the same value of allocation time through over-request cases.

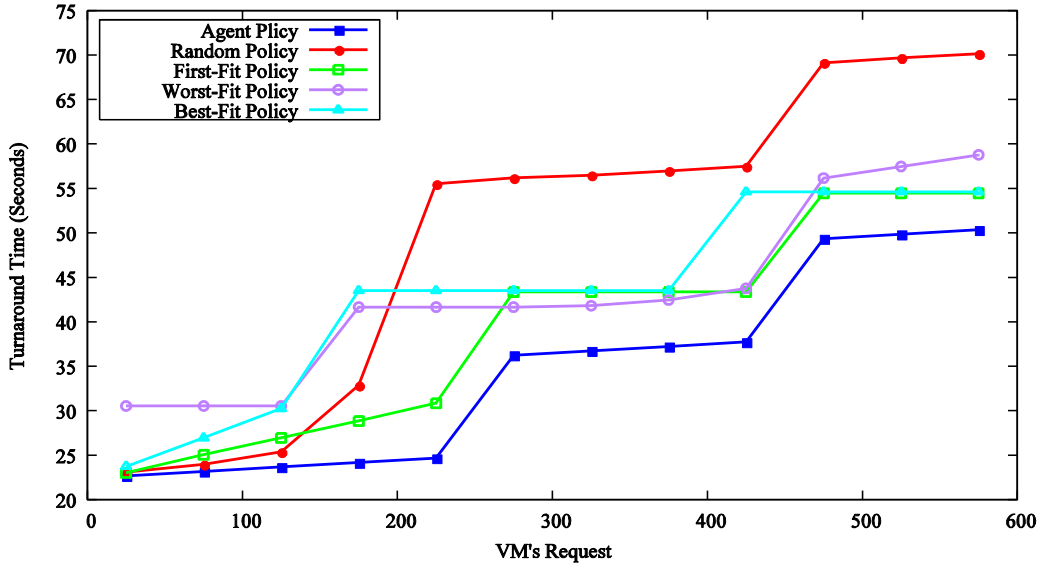

Figure 6. 27: The Turnaround Time of Algorithms vs. VMs Request.

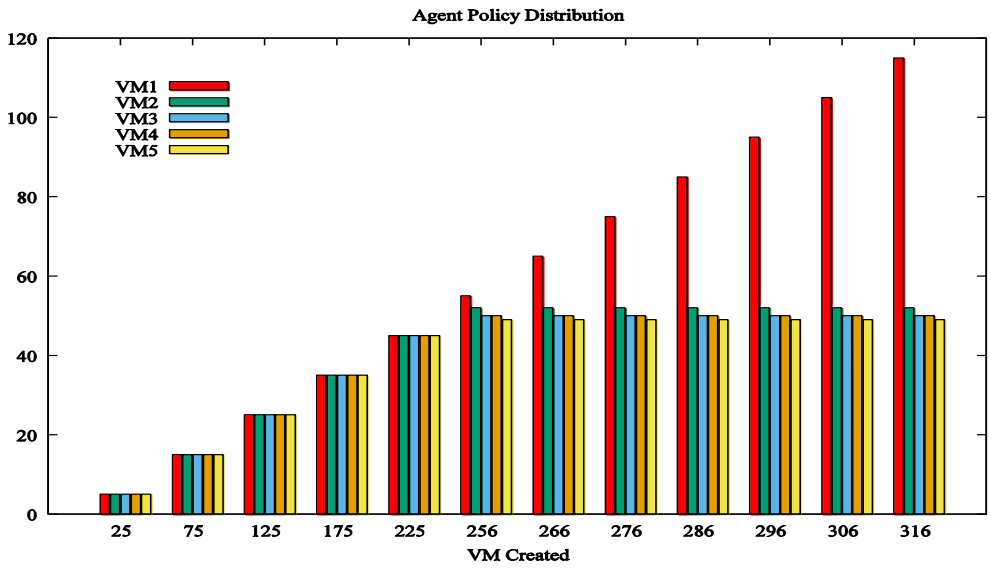

Figure 6. 28: The Distribution of Agent-based Algorithm to the Created VMs.

The Random algorithm values in allocation time through under-request cases (25 – 150 VMs) were better/lower than all Bin-Packing algorithms, but after that, a big jump happened which makes the Random algorithm values in over-request cases worse/ higher than all Bin-Packing algorithms. Agent-based again shows high efficiency in allocation time over all requesting cases, furthermore, it exhibits very steady and stable performance by the smoothly graded values of allocation time during the numerical experiments of this scenario c.f. figure 6.26.

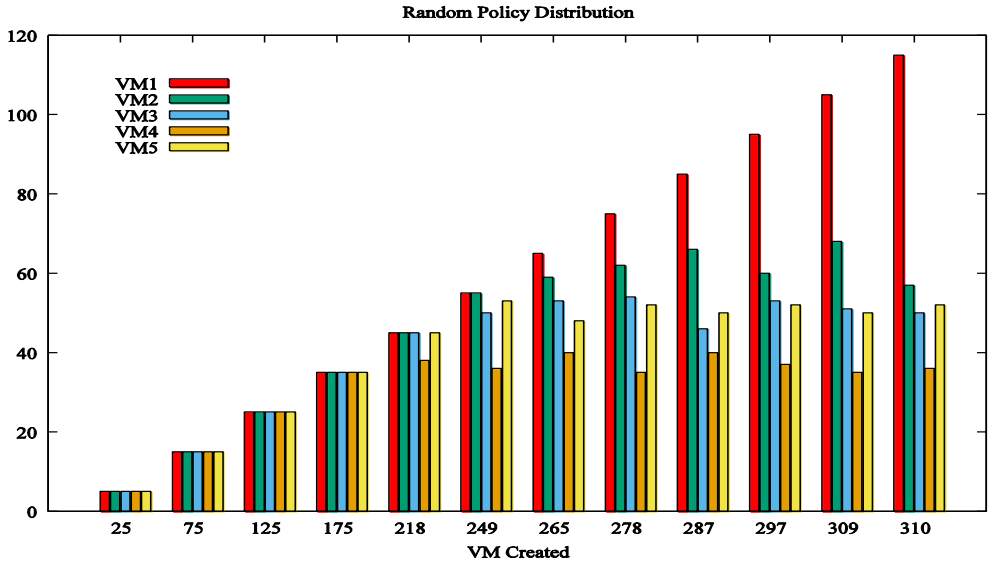

Figure 6. 29: The Distribution of Random Algorithm to the Created VMs.

Note that, turnaround time depends on allocation time, cloudlets and created VMs at least c.f. Section 5.1. Hence, we can avoid the impact of cloudlets because all scenarios use just one type of cloudlets; due to that the allocation time and created VMs have the main impact over turnaround time. Additional, the jumps/leaps in figure 6.27 come from irregular (optimal) distribution of Cloudlets over the created VMs as mentioned before.

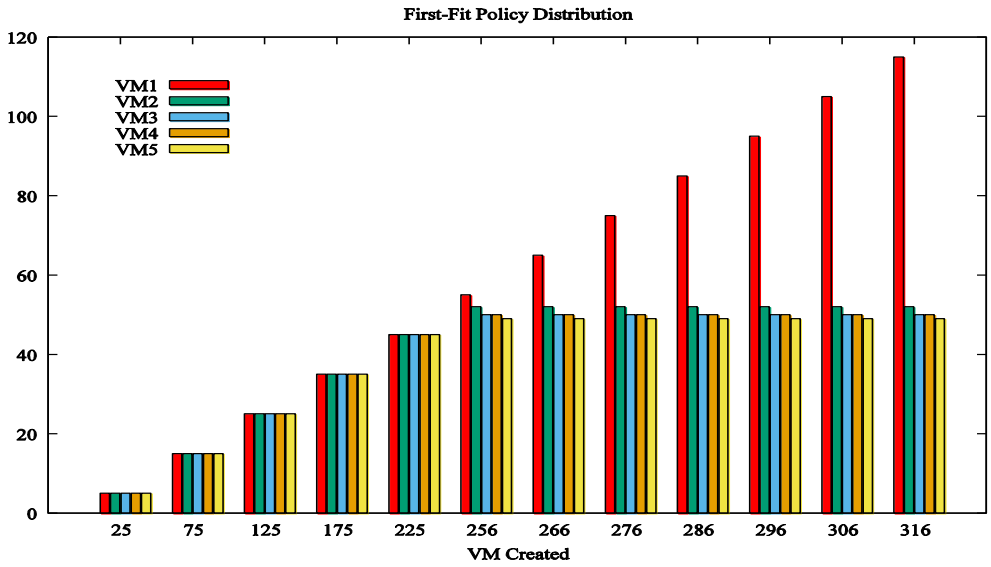

Figure 6. 30: The Distribution of the First-Fit Algorithm to the Created VMs.

The agent-based algorithm shows high efficiency in the turnaround time through all the requesting cases, further it scores the best values of turnaround time during all numerical experiments of the scenario compare with other algorithms c.f. figure 6.27. The Agent-based achieved the best values of allocation time c.f. figure 6.26 and it has a balanced distribution of VMs' type's c.f. figure 6.28, which leads Agent-based to achieve the best values of turnaround time.

A good allocation time for Random algorithm through under-request cases c.f. figure 6.26, helps it to achieve a turnaround time values better than all Bin-Packing algorithms in some cases. But the bad allocation time (after 175 VMs) of Random algorithm obviously impacts on achieving turnaround time during over-request cases. Which makes the turnaround time of all Bin-packing algorithms better than Random one.

The Bin-Packing algorithms in turnaround time figure 6.27 take the same manner in allocation time figure 2.26, where the First-Fit and Best-Fit algorithms take the leading in the beginning over the Worst-Fit one and at the end, they return to the leading again. But in

some requesting cases in the middle such as 275 and 325 VMs, the Worst-Fit algorithm takes the leading over the other two algorithms especially over Best-Fit one. Overall Bin-Packing algorithms the First-Fit one shows good performance corresponding to others two in turnaround time over all requesting cases.

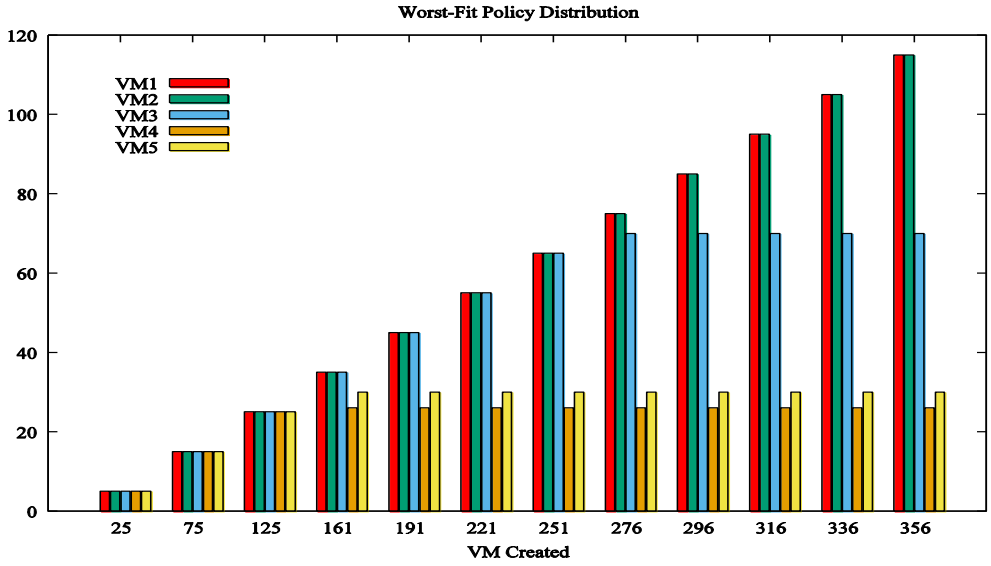

Figure 6. 31: The Distribution of Worst-Fit (Max-rest) Algorithm to the Created VMs.

According to figures 6.28-6.32, the Worst-Fit algorithm takes the first place of VM creation rate before Agent-based and First-Fit, then Random comes in third place and at last place the Best-Fit. But according to the amount of the occupied resources (especially computing power and Ram) and the balanced distribution of VMs' types, Agent-based and First-Fit algorithms take the leading over all other algorithms; due to the similarity between them in selection criteria (First-Pass-Fit and First-Fit).

The impact of hosts and VMs diversity/variety is very obvious on all algorithms during all numerical experiments of this scenario, moreover the impact of diversity/variety shows up the tiny differences between the Bin-Packing algorithms in terms of time (including allocation and turnaround) and creation process (creation rate, distribution), where the similarity of structure and verification criteria makes that very difficult through some previous scenarios.

Eventually, the good awareness about all datacenter resources which establishes by using a multi-agent system, gives the Agent-based algorithm (policy) the leading over all measurement aspects including allocation time, turnaround time, an amount of occupied resources and balanced distribution of VMs' types in this scenario. Additionally, this gives

induction about the Agent-based potential to work in a real-life cloud datacenter because it shows high efficiency and functionality with complicated scenarios like this one.

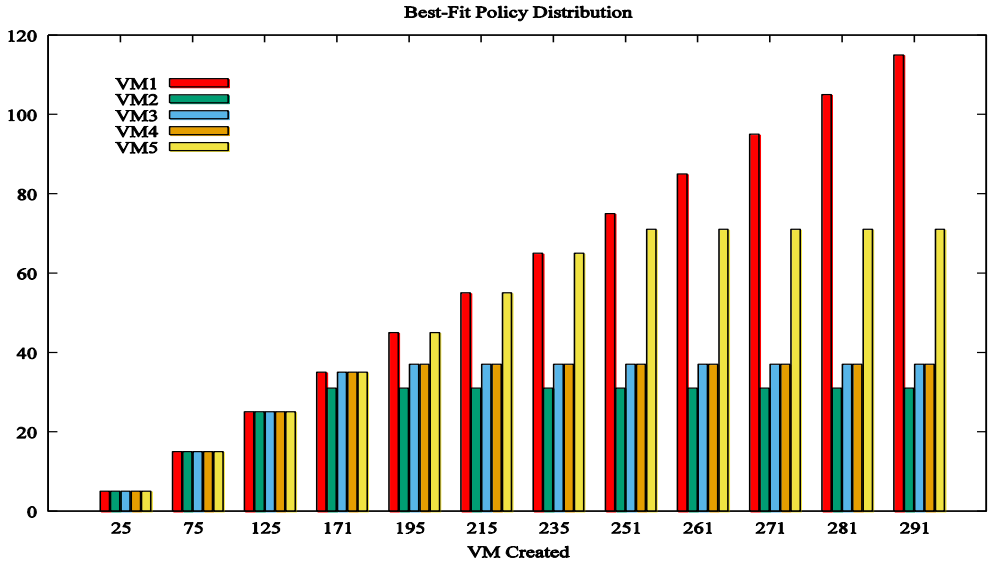

Figure 6. 32: The Distribution of Best-Fit Algorithm to the Created VMs.

This section introduced three scenarios with the Space-Shared scheduler in host level same as the last section. Next one presents a brief discussion about the comparisons between Agent-based policy and one-dimensional real-life algorithms.

6.5 Discussion

In this chapter, four of the state of the art algorithms were presented and compared with that of Agent-based policy over six scenarios, whereas, these scenarios were divided into Time-Shared scheduler and Space-Shared scheduler categories in a host scheduling level based on the CloudSim toolkit standard. Moreover, two measurement criteria are adopted to compare the results of all the algorithms of interest, these criteria are: which are time (including allocation and turnaround times) and the allocation process (VMs creation/resources occupied).

One-Dimensional design/structure (number of PEs) is adopted for the Bin-Packing algorithms in order to identify hosts in datacenter during the numerical experiments of scenarios. But the Random algorithm has no verification criteria to select the proper host in a datacenter, it was just chosen randomly. Note that the proposed Agent-based kept the same design/structure, which was used in Chapter 5 without any modification or improvement.

Using the multi-agent system gave the proposed Agent-based policy a dominance over all other algorithms during all examined scenarios in terms of time and VM allocation/placement process. In this context, the Agent-based policy clearly achieved the best allocation and turnaround times over all assessed scenarios in the two adopted categories as compared against the other algorithms. Furthermore, the Agent-based policy exhibited high performance during the VM allocation/placement process, where it created in all scenarios of the Time-Shared scheduler the highest VMs creation rate (which refers to the total number of created VMs during the numerical experiments or inter the scenario) with a balanced distribution for VMs types. Moreover, in some Space-Shared scenarios, such as scenarios 2 and 3, some of other algorithms created VMs more than the Agent-based policy. However, the latter and First-Fit algorithms achieved the highest amount of occupied resources, especially in RAM and Computational power (PEs, MIPS) with a balanced distribution for VMs types.

Due to the use of multi-agent system policy, the Agent-based algorithm constructed a full awareness about the datacenter resources, which facilitates the taking of very precise allocation decisions during the allocation process in order to select a proper host to requested VM. Consequently, in this chapter, the Agent-based policy achieved efficient results in terms of time and allocation process as compared with conventional algorithms.

Finally, the Agent-based policy showed efficient performance and functionality in high complexity cases/scenarios such as varying VMs' types and varying hosts' types; thus, the Agent-based policy has high potential to work in a real-life cloud datacenter. Moreover, the Agent-based policy exhibited superior flexibility amongst the host level schedulers (c.f., Time-Shared, Space-Shared), where it performs most efficiently and steadily under both schedulers.

6.6 Summary

This chapter carried out a detailed comparative study between the proposed Agent-based policy versus four of the state of the art algorithms, namely the Random algorithm and the three one-dimensional Bin-Packing policies, which were built in the CloudSim toolkit for the purpose of comparisons. To this end, all policies were verified/tested under both host level schedulers (Time-Shared, Space-Shared) through six scenarios, three scenarios to each scheduler for covering most of the cases in a Cloud datacenter.

Next chapter introduces another comparative study between the proposed Agent-based policy and three two-dimensional Bin-Packing algorithms.

Chapter 7 The Agent-based Policy VS. Three Bin-Packing Policies: Two-Dimensional Comparisons

7.1 Introduction

The comparative analysis approach is continuing through this chapter between Agent-based and state of the art algorithms according to the same measurement/judgment standards, for verifying the potential of using a multi-agent system in the VM allocation process among Cloud datacenter resources. In the context of that, the challenging level of comparisons is raised up, whereas the three Bin-Packing algorithms in the last chapter are built based on one-dimensional verification (checking) criteria. However, the Random algorithm is not involved in the comparisons of this chapter, because it will keep same concept and structure without any improving which means same behaving and almost the same results in Chapter 6.

The two-dimensional of verification criteria, which are adopted during Bin-Backing algorithms implementation for the comparison with Agent-based, are: (i) the *number of PEs*, and (ii) the MIPS for each PE inside the host.

The Agent-based technics (using a multi-agent system) showed high efficiency through the last chapter compared to all conventional algorithms, especially one-dimensional ones in the allocation process during broad numerical experiments over varying scenarios. Thus, this chapter presents three Bin-Packing algorithms with two-dimensional verification criteria to help them to improve their allocation progress against the Agent-based algorithm.

Note that, the Agent-based policy has the same implementation, configuration, and design which used in Chapter 4 without any modification or developing. This means it still has simple testing/verifying criteria (First-Pass-Fit) and it consists of simple make-decision agents, c.f. Section 4.5.

The three Bin-Packing algorithms are implemented and configured according to the CloudSim toolkit environment specification based on definitions in [13]; in order to compare with Agent-based algorithm among different 4 scenarios, which are divided into two categories: (i) Time-Shared scheduler [17] and (ii) Space-Shared scheduler [18]. Due to the differences in the mechanism between Time-Shared and Space-Shared schedulers; six VM

allocation policies are designed and built to perform the comparisons among two categories, this means each Bin-Packing algorithm has two versions (two policies) one for time-shared other for space-shared.

Finally, the following sections present the six two-dimensional algorithms and scenarios of Time-Shared and Space-shared schedulers, results' discussion and a brief summary.

7.2 Time-Shared Scheduler

This section presents the Time-Shared scheduler policies and scenarios through two sub-sections. Where first sub-section introduces the algorithmic steps of Bin-Packing allocation policies, which are compatible with Time-Shared scheduler techniques. The second one exposes the results of scenarios.

7.2.1 Algorithms

This sub-section presents the three algorithms (Java classes) which are implemented based on CloudSim toolkit configuration/design as VM allocation policies, so that all classes are extended from the Java abstract class called *Vm_Allocation_Policy* [74]. The algorithms as following (algorithms 7.1 – 7.3).

All policies in this sub-section follow the same concept of VM allocation policies of CloudSim toolkit, where the policy tries to find/select the proper host for requested VM based on its verification criteria and the specifications of requested VM, after that it tries to create requested VM in selected host, then return the result (True/False).

The two-dimensional structure design of verification criteria can be approved by two aspects. Firstly, verifying the current host's ability for hosting/creating the current requested VM such as point/line 5 in algorithm 7.1 First-Fit, where the verification criteria concerns in a number of PEs and MIPS of each PE. Secondly, updating the factors and variables after creating the requested VM such as points/lines 17-20 in algorithm 7.1 First-Fit, whereas the updated factors and variables involved in the verification process and represent the two-dimensional.

The time-shared scheduler is dynamic/flexible in host processing cores; this means it permits more than one VM sharing same PE if there is enough MIPS for them. Thus, Time-Shared needs own versions of allocation policies, which are different than Space-Shared ones.

Algorithm 7. 1. First-Fit 2D-Time VM Allocation Policy.

Algorithm 7.1 *First-Fit 2D-Time VM allocation policy*

Variables*: VM_Table Map, Used_PEs List, Free_PEs List, Pe_Capacity List, Available_MIPS List*

Input*: **VM** Object*

Output*: Boolean **Result***

```
1   Result=False/ required_Pes = VM.Pes/ Tries=0/max_Mips = VM.max_Mips/ total_Mips = VM.total_Mips/ idx=-1
2   If(VM is not already created)
3     Do:
4       For All (host in Host_List)
5         If(Host >= (required_Pes & max_Mips & total_Mips ))
6           idx= index of the Host;
7           Break
8         End If
9       End For
10      If (No any Host available)
11        Break
12      End If
13      Host = Host_List.get_Host(idx);
14      Result = Create_VM(Host);
15      Tries++;
16      If (Result)
17        Update:
18          VM_Table (VM)
19          Used_PEs (required_Pes)
20          Available_MIPS (total_Mips)
21        Result = true
22        Break
23      Else
24        Reset the Host:
25          Free_PEs (min_Value)
26          Available_MIPS (min_Value)
27          Pe_Capacity (min_Value)
28      End If
29    While (! Result && Tries < Free_PEs. Size);
30  End If
31  Return Result
```

Algorithm 7. 2. Worst-Fit 2D-Time VM Allocation Policy.

Algorithm 7.2 *Worst-Fit 2D-Time VM allocation policy*

Variables*: VM_Table Map, Used_PEs List, Free_PEs List, Pe_Capacity List, Available_MIPS List*

Input*: **VM** Object*

Output*: Boolean **Result***

```
1   Result=False/ required_Pes = VM.Pes/ Tries=0/max_Mips = VM.max_Mips/ total_Mips = VM.total_Mips/ idx=-1
2   If(VM is not already created)
3       Do:
4           For All (host in Host_List)
5               - Find the Host with Max-Remaining capacity of (PEs, max_Mips, total_Mips)
6               - Return the index of the Host (idx)
7           End For
8           If (No any available Host)
9               Break
10          End If
11          Host = Host_List.get_Host(idx);
12          If (Host >= (required_Pes & max_Mips & total_Mips ))
13              Result = Create_VM(Host);
14              Tries++;
15          End If
16          If (Result)
17              Update:
18                  VM_Table (VM)
19                  Used_PEs (required_Pes)
20                  Available_MIPS (total_Mips)
21              Result = true
22              Break
23          Else
24              Reset the Host:
25                  Free_PEs (min_Value)
26                  Available_MIPS (min_Value)
27                  Pe_Capacity (min_Value)
28          End If
29      While (! Result && Tries < Free_PEs. Size);
30  End If
31  Return Result
```

Algorithm 7. 3. Best-Fit 2D-Time VM Allocation Policy.

Algorithm 7.3 *Best-Fit 2D-Time VM allocation policy*

Variables*: VM_Table Map, Used_PEs List, Free_PEs List, Pe_Capacity List, Available_MIPS List*
Input*: **VM** Object*
Output*: Boolean **Result***

```
1   Result=False/ required_Pes = VM.Pes/ Tries=0/max_Mips = VM.max_Mips/
    total_Mips = VM.total_Mips/ idx=-1/ ratio =0
2   If(VM is not already created)
3     Do:
4       For All (host in Host_List)
5         If(Host >= (required_Pes & max_Mips & total_Mips ))
6           Temp_ratio = Calculate remaining capacity of the Host (Pes, MIPS)
            after add the VM (Pes, MIPS).
7           If (Temp_ratio >= ratio)
8             idx= index of the host;
9             ratio = Temp_ratio;
10          End If
13        End If
14      End For
15      If (No any Host available)
16        Break
17      End If
18      Host = Host_List.get_Host(idx);
19      Result = Create_VM(Host);
20      Tries++;
21      If (Result)
22        Update:
23          VM_Table (VM)
24          Used_PEs (required_Pes)
25          Available_MIPS (total_Mips)
26        Result = true
27        Break
28      Else
29        Reset the Host:
30          Free_PEs (min_Value)
31          Available_MIPS (min_Value)
32          Pe_Capacity (min_Value)
33      End If
34    While (! Result && Tries < Free_PEs. Size);
35  End If
36  Return Result
```

Note that, the update manner of any algorithm to its factors and variables indicates the general fashion (dynamic/static) of the allocation policy, so that if the update block does not lock/reserve host PE for just one VM and allow the sharing with another VM according to the available MIPS; which means this algorithm belongs to Time-Shared, otherwise it belongs to the Space-Shared, c.f. algorithms 7.1-7.3.

Finally, the next sub-section will introduce two scenarios to verify the two-dimensional Bin-Packing Algorithms performance against Agent-based policy.

7.2.2 The Scenarios

This sub-section aims to test/verify all algorithms through complex situations (real life) such as VMs variety and host variety under the impact of Time-Shared scheduler. Thus, it presents two complex scenarios and avoids simple scenarios like scenario 6.3.1 in the last chapter, moreover Agent-based domination/advantage over Bin-Packing algorithms is approved in simple scenarios through the last two chapters.

7.2.2.1 Scenario I: VMs Variety

This scenario is devoted to test/verify the impact of VMs variety (more than one type) on the efficiency of all algorithms in terms of time (allocation, turnaround) and allocation VMs (placement, distribution), where it uses three VM types.

Note that, this scenario imitates the same situation of scenario 6.3.3, but Bin-Packing algorithms have advanced verification criteria (two-dimensional) here.

Table 7. 1. The VMs Features.

Feature	*VM 1*	*VM 2*	*VM 3*
Image size	10000 MB	10000 MB	10000 MB
RAM	1 GB	2 GB	3 GB
MIPS	1,000	1,500	1,000
Bandwidth	1000	2000	2000
Number of Pes	1	2	1
VMM	XEN	XEN	XEN

Table 7. 2. The Hosts Features.

Feature	*Host 1*	*Host 2*	*Host 3*
Number of Pes	4	5	4
MIPS	2000	1500	3000
RAM	8 GB	10 GB	10 GB
Bandwidth	10,000 MHz	10,000 MHz	10,000 MHz
Storage	1 Terabyte	1 Terabyte	1 Terabyte

Assumptions and conditions of the scenario: First, using three types of VMs as shown in table 7.1. Second, using three types of Hosts as shown in table 7.2, Third, the number of hosts is fixed (120 hosts); divided into three groups equally, 40 hosts for each type. Forth, the number of requested VMs is varied, but the requesting amount of each VM type is equal.

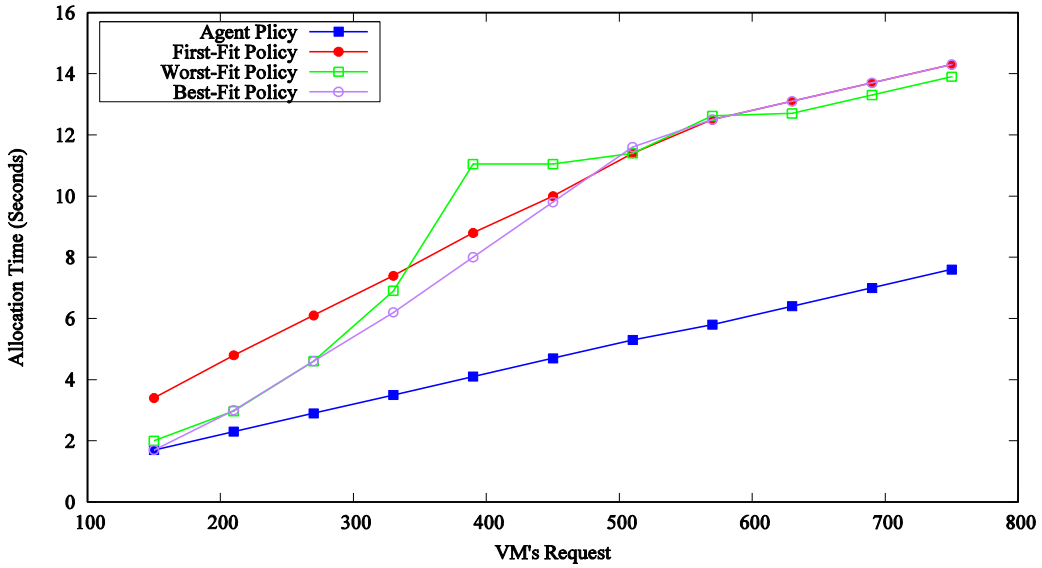

Figure 7. 1: The Allocation Time of Algorithms with VMs Variety.

Agent-based algorithm exhibits high-performance efficiency with VMs variety compared with other algorithms during all requesting cases (under-request, over-request) over the Allocation Time, also the performance is smooth and gradually from under-request cases (200 VMs) to over-request cases (700 VMs) c.f. figure 7.1. In addition, Agent-based shows high stability during the scenario experiments under the impact of VMs' variety, which is opposite of Bin-Packing algorithms.

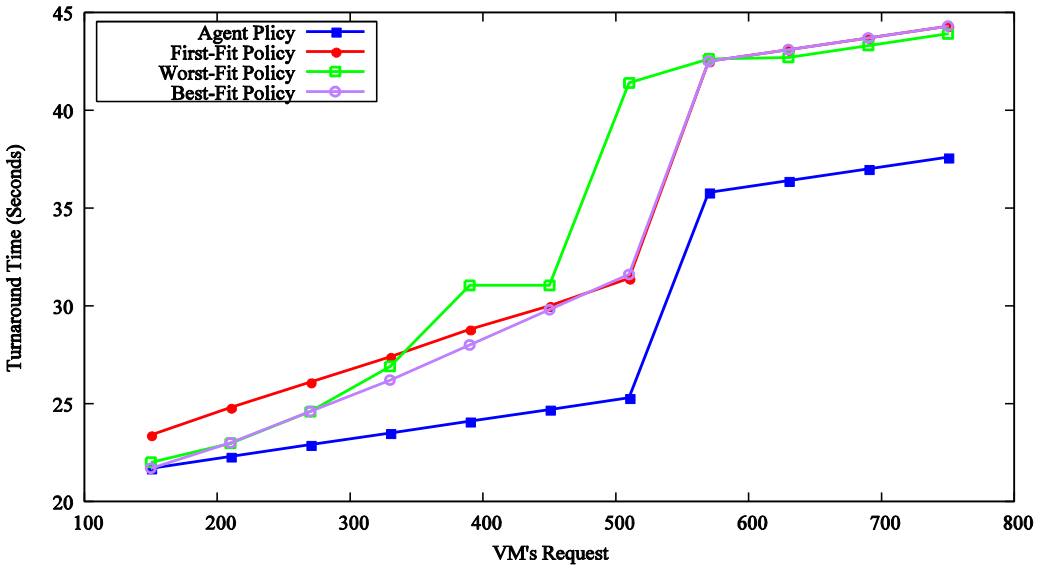

Figure 7. 2: The Turnaround Time of Algorithms with VMs Variety.

Due to the similarity of structure and verification criteria between the Bin-Packing (First, Worst, Best), they achieved almost the same allocation time in over-request cases (Max-boundaries) especially First-Fit and Best-Fit. But, different allocation times are achieved by Bin-Packing algorithms in under-request cases, this means the VMs variety has a clear impact on Bin-Packing algorithms especially in under-request cases c.f. figure 7.1.

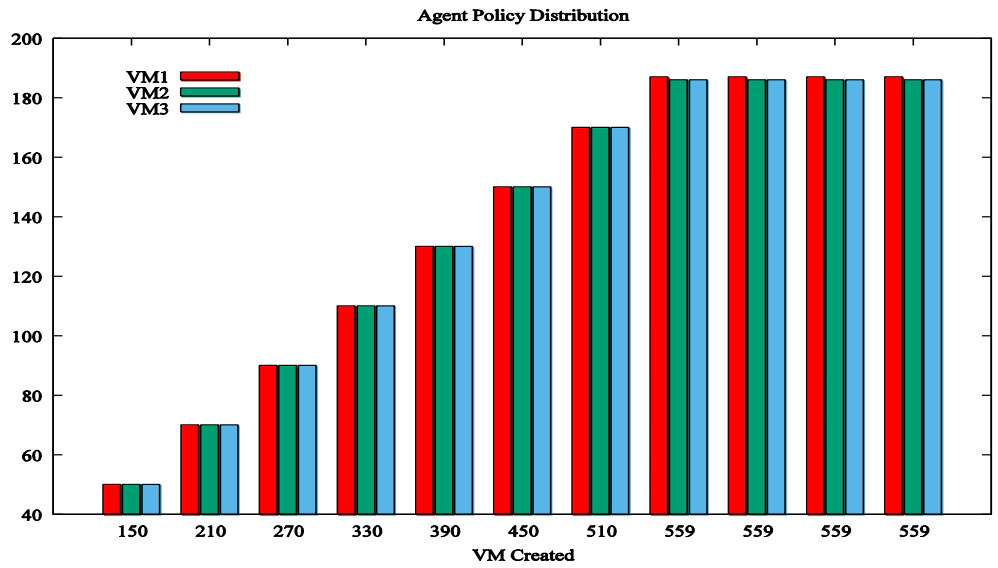

Figure 7. 3: The Distribution of Agent-based policy to the Created VMs.

In turnaround time, Agent-based remains achieving best times over all requesting cases (under, over) compared with other algorithms, especially in over-request cases such as 700 requested VMs; duo to the leading of Agent-based in allocation time and rate of created VMs with First-Fit c.f. figures 7.3-7.6. Noticeably, the gap between First-Fit algorithm and Agent-based almost stable or fixed during turnaround time because almost there is no difference between them in the created VMs c.f. figures 7.3 and 7.4.

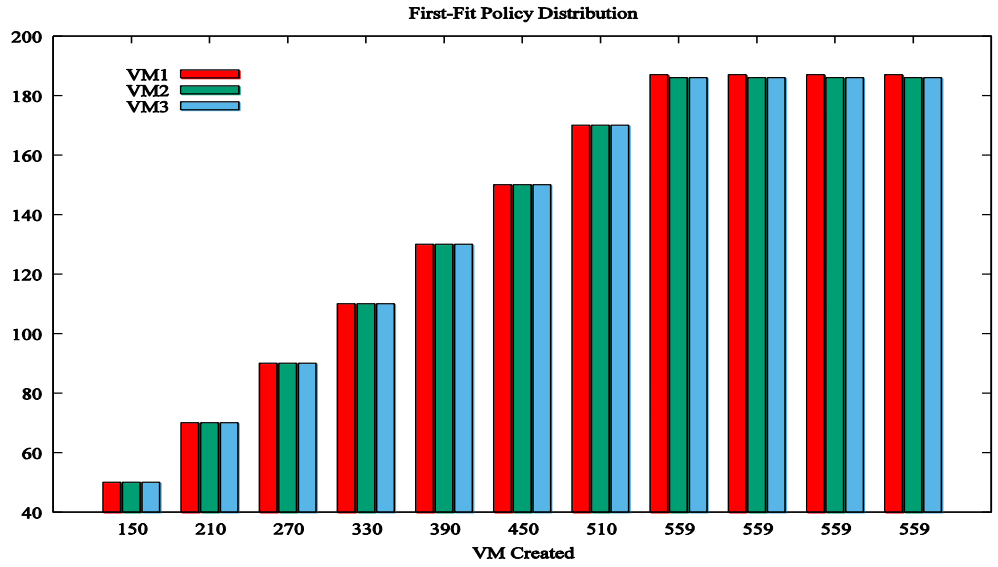

Figure 7. 4: The Distribution of First-Fit policy to the Created VMs.

The stairs shape of all algorithm curves (lines) in turnaround time figure comes from the irregular distribution of Cloudlet over the created VMs. Whereas, in the regular distribution the created VMs equal the requested VMs (optimal distribution), which means the turnaround line graded smoothly corresponding to the requested VMs. Thus, the steps or leaps appear (show up) when the gap between the created VMs and requested VMs increase (non-optimal distribution) in some requesting cases c.f. figure 7.2.

The First-Fit and Best-Fit algorithms achieved almost same turnaround times and same line shape especially in over-request cases, due to the allocation time values were very close under-request cases or same in over-request cases and the distribution of created VMs is a balance. But, the turnaround line of Worst-Fit is not stable and smoothly (many jumps) like other algorithms, because the creation rate and distribution of created VMs are not stable and smoothly c.f. figures 7.2 and 7.5.

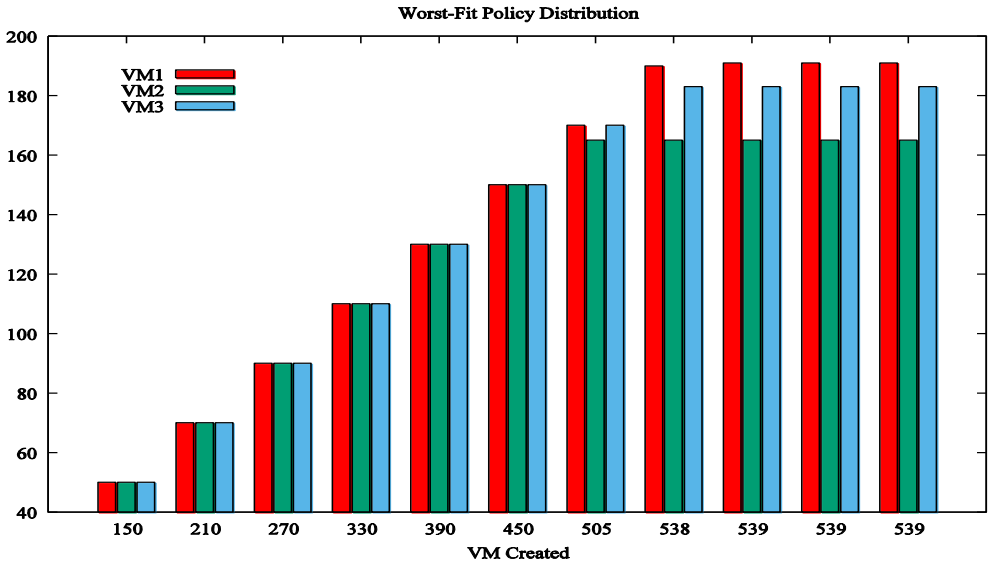

Figure 7. 5: The Distribution of Worst-Fit policy to the Created VMs.

According to the Distribution figures (7.3-7.6), there is a gap in the VMs creation rate between: firstly, Agent-based and First-Fit algorithms which created the same amount of VMs at the end. And secondly, the other algorithms that created a different amount of VMs. Agent-based and First-Fit have achieved the highest creation rate with well-balanced distribution amongst the VMs types compare with other algorithms especially the Worst-Fit one.

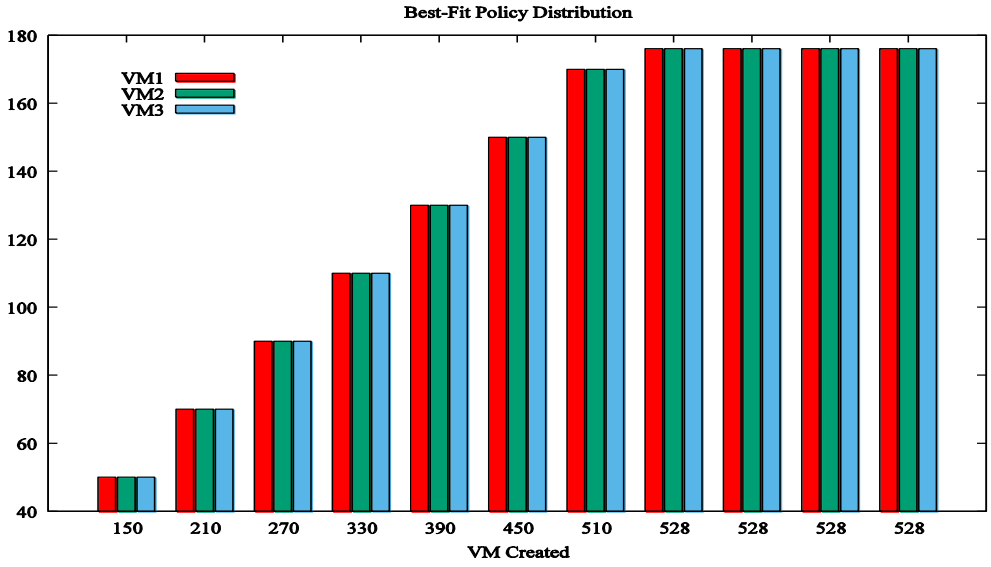

Figure 7. 6: The Distribution of Best-Fit policy to the Created VMs.

Note that, the similarity in verification criteria structure between Agent-based (First-Pass-Fit) and First-Fit, helps a First-Fit to achieve the same VMs creation rate and keep the same distance from Agent-based through turnaround time, c.f. figures 7.2-7.4.

Finally, the Agent-based policy has exhibited high efficiency corresponds to other algorithms during this scenario in terms of time including allocation and turnaround times c.f. figures 7.1 and 7.2, and allocation process including the VMs creation rate and VMs types distribution with the First-Fit c.f. figures 7.3-7.6.

7.2.2.2 Scenario II: VMs Diversity vs. Hosts Diversity

The scenario aims to imitate the real-life situation, where the real cloud datacenter has a variety of hosts' types and the VMs are varying through the requesting process. In terms of that, this scenario has five host types in the datacenter and five VM types are requested through the requesting cases in numerical experiments.

Table 7. 3. The VMs Features.

Feature	*VM 1*	*VM 2*	*VM 3*	*VM 4*	*VM 5*
Image size	10000 MB	10000 MB	10000 MB	10000 MB	10000 MB
RAM	1 GB	2 GB	3 GB	3 GB	4 GB
MIPS	900	1,000	2,000	1,000	1,500
Bandwidth	1000	2000	2000	2000	2000
Number of Pes	1	1	2	3	3
VMM	XEN	XEN	XEN	XEN	XEN

Table 7. 4. The Hosts Features.

Feature	*Host 1*	*Host 2*	*Host 3*	*Host 4*	*Host 5*
Number of Pes	3	5	4	4	3
MIPS	2000	1500	3000	2000	3000
RAM	6 GB	8 GB	8 GB	10 GB	8 GB
Bandwidth	10,000 MHz	10,000 MHz	10,000 MHz	10,000 MHz	10,000 MHz
Storage	1 Terabyte	1 Terabyte	1 Terabyte	1 Terabyte	1 Terabyte

Conditions and assumptions: First, using five types of VMs as shown in table 7.3. Second, using five types of Hosts as shown in table 7.4, Third, the number of hosts is fixed (200 hosts); divided into five groups equally, 40 hosts for each type. Forth, the number of requested VMs is varied, but the requesting amount of each VM type is equal.

The impact of Hosts and VMs diversity is very obvious over the allocation time (allocation process) on Bin-Packing algorithms, where the differences between all Bin-Packing algorithms appear exactly during the under-request cases. The First-Fit during all request cases (under, over) shows stable functionality and scored gradually values of allocation time until the end, but the values were higher than others Bin-Packing algorithms through under-request cases, c.f. figure 7.7.

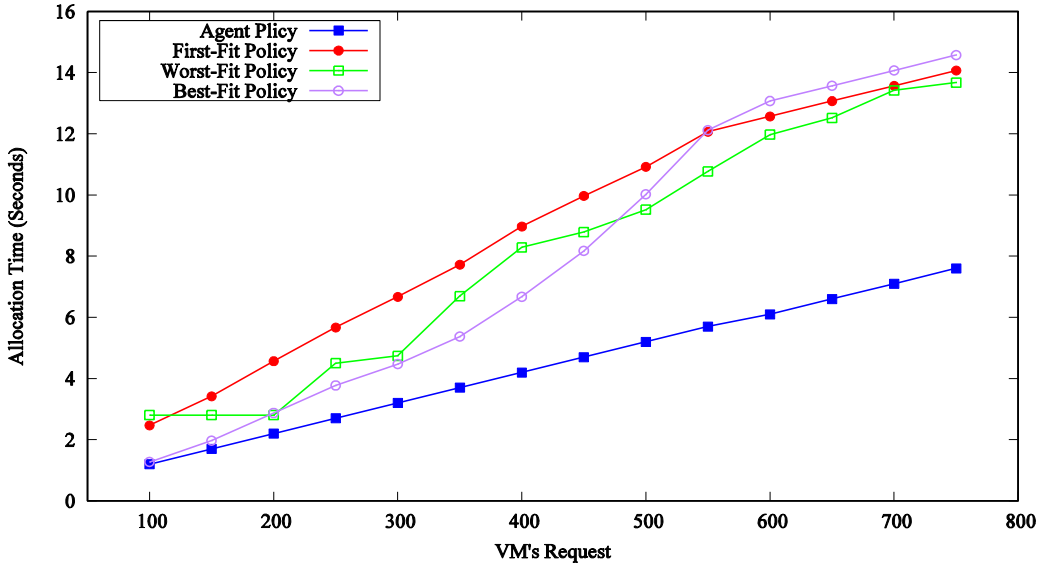

Figure 7. 7: The Allocation Time For Algorithms.

The Best-Fit algorithm in figure 7.7, score very close values to the Agent-based in the beginning, then the values of allocation time started growing up during under-request cases of VMs. But, in over-request cases (after 550 VMs) scored the higher values until the end, which means it started struggling with the variety (hosts, VMs) through over-request cases. The Worst-fit shows unstable performance during all request cases, whereas its allocation time curve takes the zigzag shape. Despite that, Worst-Fit takes the second place in under-request cases like 250 VMs and first place in over-request cases like 700 VMs amongst the Bin-Packing algorithms c.f. figure 7.7.

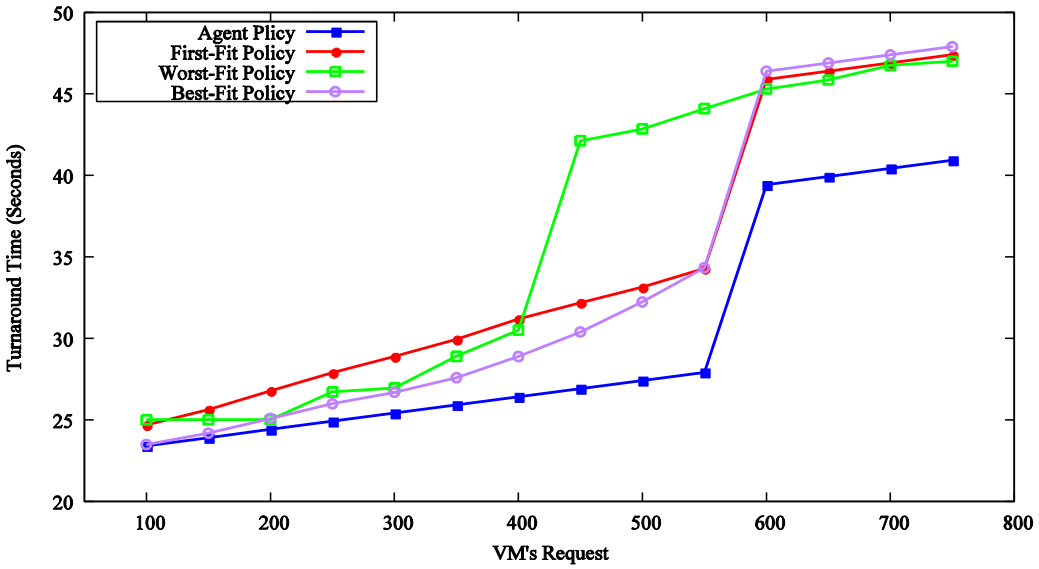

Figure 7. 8: The Turnaround Time For Algorithms.

Agent-based again shows again high efficiency in allocation time over all requesting cases, furthermore, it exhibits very steady and stable performance by the smoothly graded values of allocation time during the numerical experiments of this scenario c.f. figure 7.7.

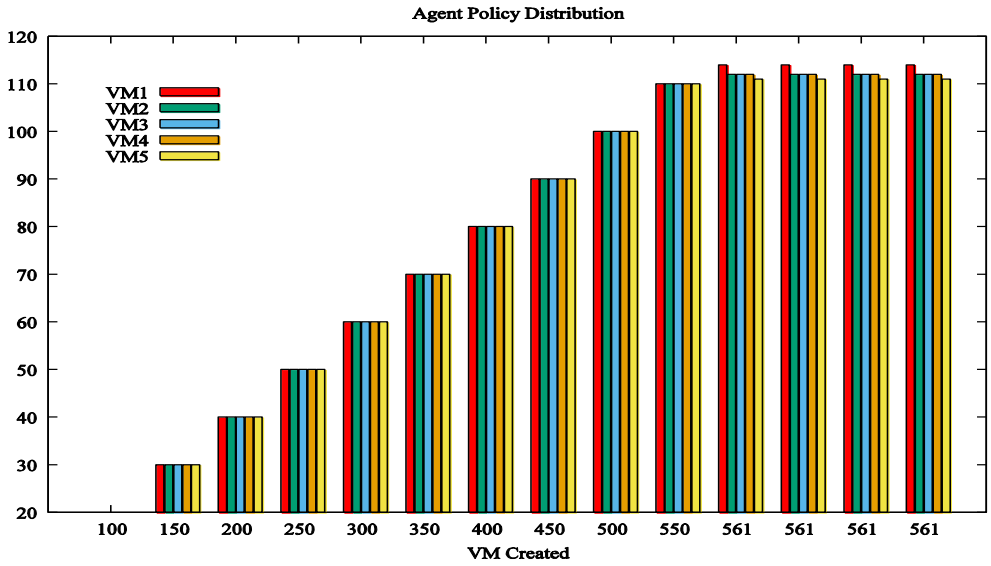

Figure 7. 9: The Distribution of Agent-based Algorithm to the Created VMs.

Turnaround time depends on allocation time, cloudlets and created VMs at least c.f. Section 5.1. Hence, we can avoid the impact of cloudlets because all scenarios use just one type of cloudlets; due to that the allocation time and created VMs (occupied resources) have the main impact over turnaround time.

The agent-based algorithm shows high efficiency in the turnaround time through all the requesting cases, further it scores the best values of turnaround time during all numerical experiments of the scenario compare with other algorithms c.f. figure 7.8. The Agent-based achieved the best values of allocation time c.f. figure 7.7, and it occupied with First-Fit the maximum amount of resources in PEs and MIPS, with a balanced distribution of VMs' type's c.f. figure 7.9, that leads Agent-based to achieve the best values of turnaround time.

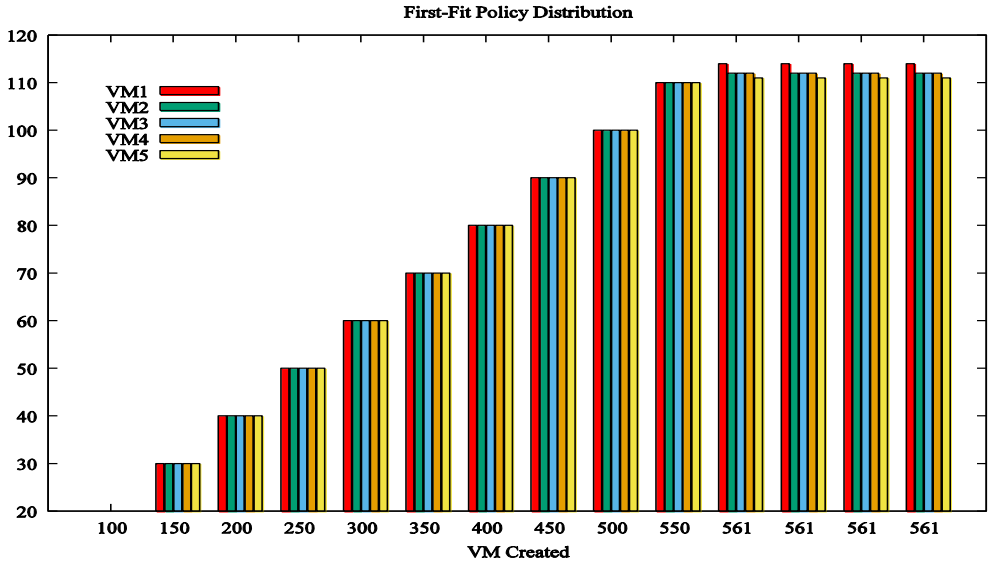

Figure 7. 10: The Distribution of the First-Fit Algorithm to the Created VMs.

The Bin-Packing algorithms in turnaround time figure 7.8 take the same manner in allocation time figure 7.7, where the Best-Fit algorithm takes the leading in the beginning over the Worst-Fit and First-Fit but at the end, it achieved the worst time. Also, the Worst-Fit algorithm takes the leading over the other two algorithms, especially in over-request cases. Overall Bin-Packing algorithms the First-Fit one shows steady and stable performance corresponding to others two in turnaround time over all requesting cases and it almost keeps the distance with Agent-based fixed, c.f. figure 7.8.

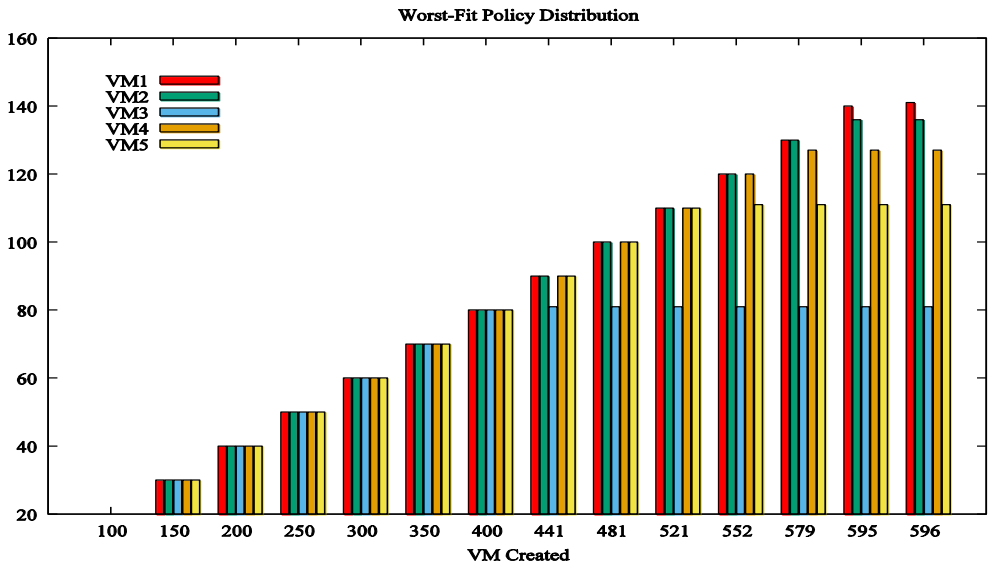

Figure 7. 11: The Distribution of Worst-Fit Algorithm to the Created VMs.

The jumps/leaps in figure 7.8 come from irregular (optimal) distribution of Cloudlets over the created VMs as mentioned before. Agent-based and First-Fit have almost a regular/steady creation process over all request cases with a balanced distribution of VMs' types c.f. figures 7.9, 7.10, this helps Agent-based to make smoothly/steady turnaround time curve with one leap/jump compare with others.

According to figures 7.9-7.12, the Worst-Fit algorithm takes the first place of the VM creation rate, but the distribution of VMs' types is an average balance, and the last place of the VM creation rate went to Best-Fit one with good balance distribution of VMs' types. However, The Agent-based and First-Fit algorithm take the second place of VMs creation rate, but they occupied PEs and MIPS more than the Worst-Fit because they give smoothly balance the distribution of VMs' types.

The impact of hosts and VMs diversity/variety is very obvious on all algorithms during all numerical experiments of this scenario, moreover the impact of diversity/variety shows up the differences between the Bin-Packing algorithms in terms of time (including allocation and turnaround) and creation process (creation rate, distribution), where the two-dimensional verification criteria makes that very clear compared with last chapter scenarios.

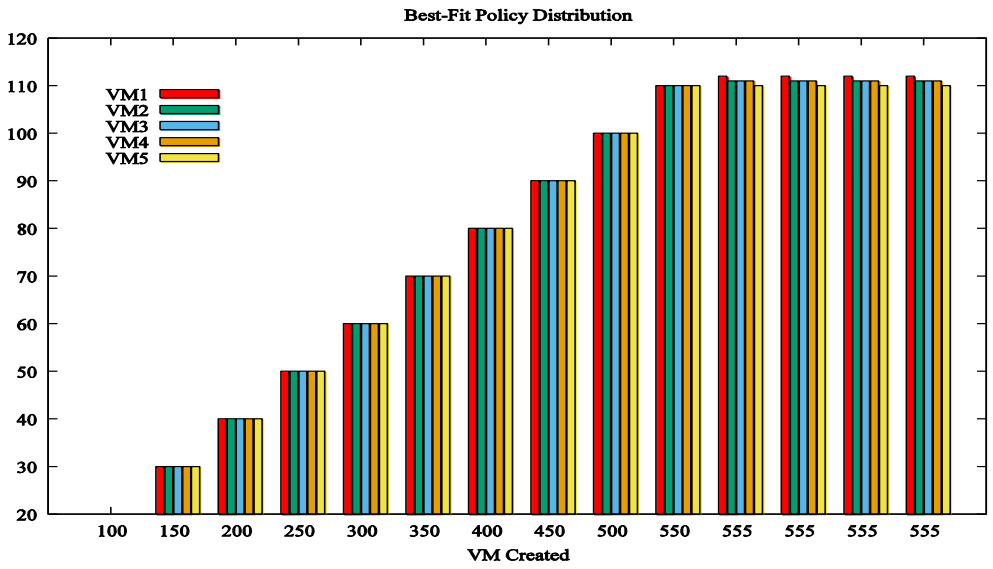

Figure 7. 12: The Distribution of Best-Fit Algorithm to the Created VMs.

Finally, the good awareness about all datacenter resources which built by using a multi-agent system, gives the Agent-based algorithm (policy) the leading over all measurement aspects including allocation time, turnaround time, occupied resources (PEs and MIPS) and balanced distribution of VMs' types in this scenario. Also, this gives induction about the Agent-based potential to work in a real cloud datacenter because it shows high efficiency and functionality with complicated scenarios.

After introducing two scenarios through this sub-section, the next section will introduce the Space-Shared Bin-Packing versions and scenarios.

7.3 Space-Shared Scheduler

The section presents the Space-Shared scheduler policies and scenarios through two sub-sections like the last section. Where first sub-section introduces the algorithmic steps of Bin-Packing allocation policies. Which are compatible with Space-Shared scheduler techniques. The second one exhibits the results of scenarios.

7.3.1 Algorithms

This sub-section introduces the Bin-Packing algorithms (Java classes) which are implemented based on CloudSim toolkit configuration/design as VM allocation policies. So that all classes are extended from the Java abstract class called *Vm_Allocation_Policy* [74]. Further, all policies

follow the same concept of VM allocation policies of CloudSim toolkit, which is mentioned in Sub-Section 7.2.1. The algorithms as following.

Algorithm 7. 4. The First-Fit 2D-Space VM Allocation Policy.

Algorithm 7.4 *First-Fit 2D-Space VM allocation policy*

Variables*: VM_Table Map, Used_PEs List, Free_PEs List, Pe_Capacity List, Available_MIPS List*
Input*: **VM** Object*
Output*: Boolean **Result***

```
1   Result=False/ required_Pes = VM.Pes/ Tries=0/max_Mips = VM.max_Mips/ total_Mips = VM.total_Mips/ idx=-1
2   If(VM is not already created)
3     Do:
4       For All (host in Host_List)
5         If(Host >= (required_Pes & max_Mips & total_Mips))
6           idx= index of the Host;
7           Break
8         End If
9       End For
10      If (No any Host available)
11        Break
12      End If
13      Host = Host_List.get_Host(idx);
14      Result = Create_VM(Host);
15      Tries++;
16      If (Result)
17        Update:
18          VM_Table (VM)
19          Used_PEs (required_Pes)
            Free_PEs (required_Pes)
20          Available_MIPS (total_Mips)
21        Result = true
22        Break
23      Else
24        Reset the Host:
25          Free_PEs (min_Value)
26          Available_MIPS (min_Value)
27          Pe_Capacity (min_Value)
28      End If
29    While (! Result && Tries < Free_PEs. Size);
30  End If
31  Return Result
```

Algorithm 7. 5. The Worst-Fit 2D-Space VM Allocation Policy.

Algorithm 7.5 *Worst-Fit 2D-Space VM allocation policy*

Variables*: VM_Table Map, Used_PEs List, Free_PEs List, Pe_Capacity List, Available_MIPS List*
Input*: **VM** Object*
Output*: Boolean **Result***

```
1   Result=False/ required_Pes = VM.Pes/ Tries=0/max_Mips = VM.max_Mips/ total_Mips = VM.total_Mips/ idx=-1
2   If(VM is not already created)
3     Do:
4       For All (host in Host_List)
5         - Find the Host with Max-Remaining capacity of (PEs, max_Mips, total_Mips)
6         - Return the index of the Host (idx)
7       End For
8       If (No any Host available)
9         Break
10      End If
11      Host = Host_List.get_Host(idx);
12      If (Host >= (required_Pes & max_Mips & total_Mips ))
13        Result = Create_VM(Host);
14        Tries++;
15      End If
16      If (Result)
17        Update:
18          VM_Table (VM)
19          Used_PEs (required_Pes)
            Free_PEs (required_Pes)
20          Available_MIPS (total_Mips)
21        Result = true
22        Break
23      Else
24        Reset the Host:
25          Free_PEs (min_Value)
26          Available_MIPS (min_Value)
27          Pe_Capacity (min_Value)
28      End If
29    While (! Result && Tries < Free_PEs. Size);
30  End If
31  Return Result
```

Algorithm 7. 6. The Best-Fit 2D-Space VM Allocation Policy.

Algorithm 7.6 *Best-Fit 2D-Space VM allocation policy*

Variables*: VM_Table Map, Used_PEs List, Free_PEs List, Pe_Capacity List, Available_MIPS List*
Input*: **VM** Object*
Output*: Boolean **Result***

```
1   Result=False/ required_Pes = VM.Pes/ Tries=0/max_Mips = VM.max_Mips/ total_Mips = VM.total_Mips/ idx=-1/ ratio =0
2   If(VM is not already created)
3     Do:
4       For All (host in Host_List)
5         If(Host >= (required_Pes & max_Mips & total_Mips ))
6           Temp_ratio = Calculate remaining capacity of the Host (Pes, MIPS)
            after add the VM (Pes, MIPS).
7           If (Temp_ratio >= ratio)
8             idx= index of the host;
9             ratio = Temp_ratio;
10          End If
13        End If
14      End For
15      If (No any Host available)
16        Break
17      End If
18      Host = Host_List.get_Host(idx);
19      Result = Create_VM(Host);
20      Tries++;
21      If (Result)
22        Update:
23          VM_Table (VM)
24          Used_PEs (required_Pes)
25          Free_PEs (required_Pes)
26          Available_MIPS (total_Mips)
27        Result = true
28        Break
29      Else
30        Reset the Host:
31          Free_PEs (min_Value)
32          Available_MIPS (min_Value)
33          Pe_Capacity (min_Value)
34      End If
35    While (! Result && Tries < Free_PEs. Size);
36  End If
37  Return Result
```

The two-dimensional structure design of verification criteria can be approved through two pseudo-code blocks in any algorithm, which are the verifying and updating blocks, c.f. Sub-Section 7.2.1.

The space-shared scheduler is static/fixed in host processing cores; this means it prevents more than one VM sharing same PE. Thus, it needs own versions of allocation policies, which are different than Time-Shared ones.

Note that, the update manner of any algorithm to its factors and variables indicates the general fashion (dynamic/static) of the allocation policy, so that if the update block locks/reserves host PE for just one VM and prevents the sharing with another VM; this means the algorithm belongs to Space-Shared, otherwise it belongs to Time-Shared, c.f. algorithms 7.4-7.6.

Finally, the next sub-section will introduce two scenarios to verify the two-dimensional Bin-Packing Algorithms performance against Agent-based policy.

7.3.2 The Scenarios

This sub-section similar to Sub-Section 7.2.2 whereas it aims to test/verify all algorithms through complex situations like VMs variety and host variety under the impact of the Space-Shared scheduler. Thus, it presents two complex scenarios and avoids simple scenarios like scenario 6.4.1 in the last chapter, because the Agent-based domination/advantage over Bin-Packing algorithms is approved in simple scenarios through last two chapters.

7.3.2.1 Scenario I: VMs Variety

The scenario is devoted to test/verify the impact of VMs variety on the efficiency of all algorithms in terms of time (allocation, turnaround) and allocation VMs (placement, distribution), where it uses three VM types.

Note that, this scenario imitates the same situation of scenario 6.4.3, but Bin-Packing algorithms have advanced verification criteria (two-dimensional) here.

Table 7. 5. The VMs Features.

Feature	*VM 1*	*VM 2*	*VM 3*
Image size	10000 MB	10000 MB	10000 MB
RAM	2 GB	2 GB	3 GB
MIPS	1,000	1,500	1,000
Bandwidth	1000	2000	2000
Number of Pes	1	2	1
VMM	XEN	XEN	XEN

The scenario assumptions and conditions: First, using three types of VMs as shown in table 7.5. Second, using three types of Hosts as shown in table 7.6, Third, the number of hosts is fixed (120 hosts); divided into three groups equally, 40 hosts for each type. Forth, the number of requested VMs is varied, but the requesting amount of each VM type is equal.

Table 7. 6. The Hosts Specifications.

Feature	*Host 1*	*Host 2*	*Host 3*
Number of Pes	4	5	4
MIPS	2000	1500	3000
RAM	6 GB	8 GB	8 GB
Bandwidth	10,000 MHz	10,000 MHz	10,000 MHz
Storage	1 Terabyte	1 Terabyte	1 Terabyte

Agent-based algorithm exhibits high-performance efficiency with VMs variety compared with other algorithms during all requesting cases (under-request, over-request) over the Allocation Time, also the performance is smooth and gradually from under-request cases (100 VMs) to over-request cases (480 VMs) c.f. figure 7.13. Moreover, Agent-based shows high stability during the scenario experiments under the impact of VMs' variety, which is opposite of all Bin-Packing algorithms.

Despite to the similarity of structure and verification criteria between the Bin-Packing algorithms, they achieved different allocation time in all request cases. This means the VMs variety has a clear impact on Bin-Packing algorithms also with using Space-Shared c.f. figure 7.13.

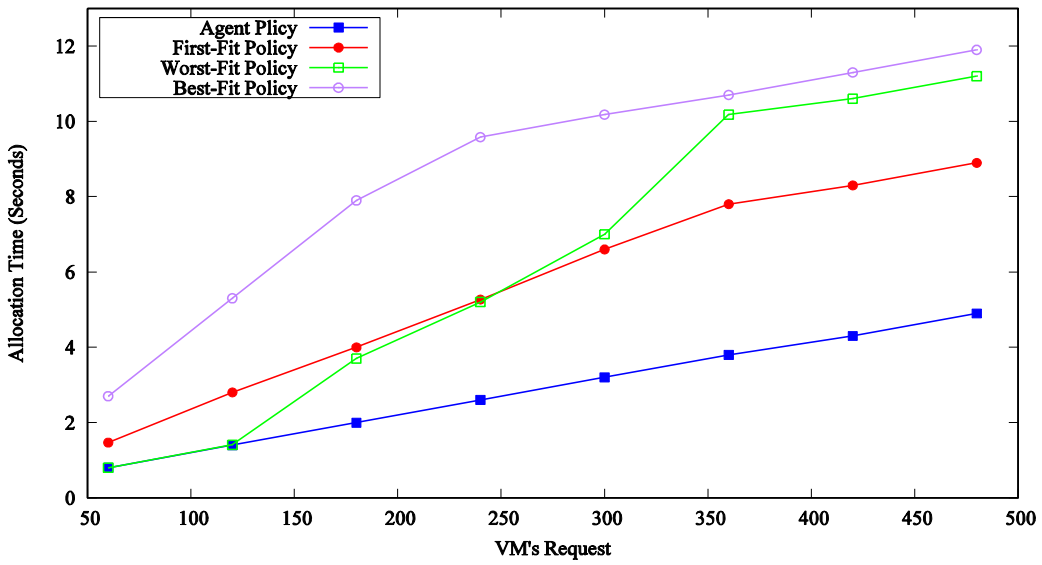

Figure 7. 13: The Allocation Time of Algorithms.

In turnaround time, Agent-based remains achieving best times over all requesting cases (under, over) compared with other algorithms, especially in over-request cases such as 480 requested VMs; duo to the leading of Agent-based in allocation time and rate of created VMs with First-Fit c.f. figures 7.15-7.18. Noticeably, the distance between First-Fit algorithm and Agent-based almost stable or fixed during turnaround time because both of them create the same amount of VMs c.f. figures 7.15 and 7.16.

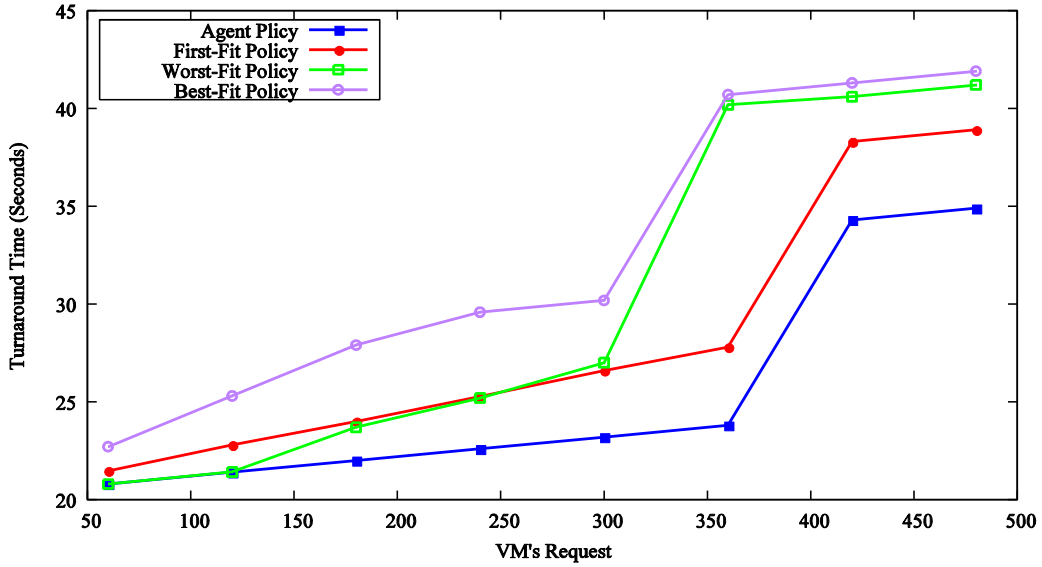

Figure 7. 14: The Turnaround Time of Algorithms.

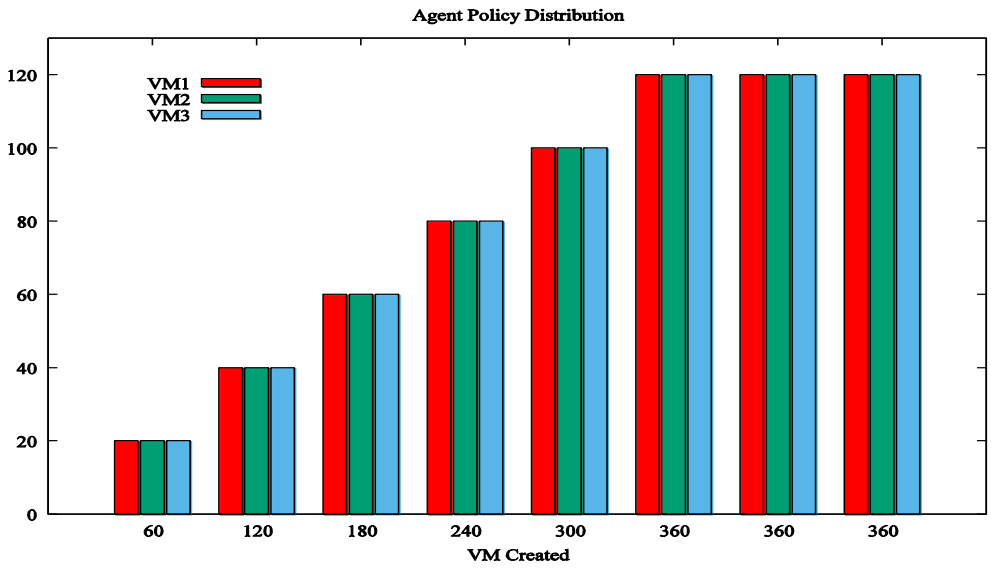

Figure 7. 15: The Distribution of Agent-based Algorithm to the Created VMs.

The stairs shape of all algorithm curves/lines in turnaround time comes from the irregular distribution of Cloudlet over the created VMs. In the regular/optimal distribution, the created VMs equal the requested VMs, which means the turnaround line graded smoothly corresponding to the requested VMs. Thus, the steps or leaps appear (show up) when the gap between the created VMs and requested VMs increase (non-optimal distribution) in some requesting cases c.f. figure 7.14.

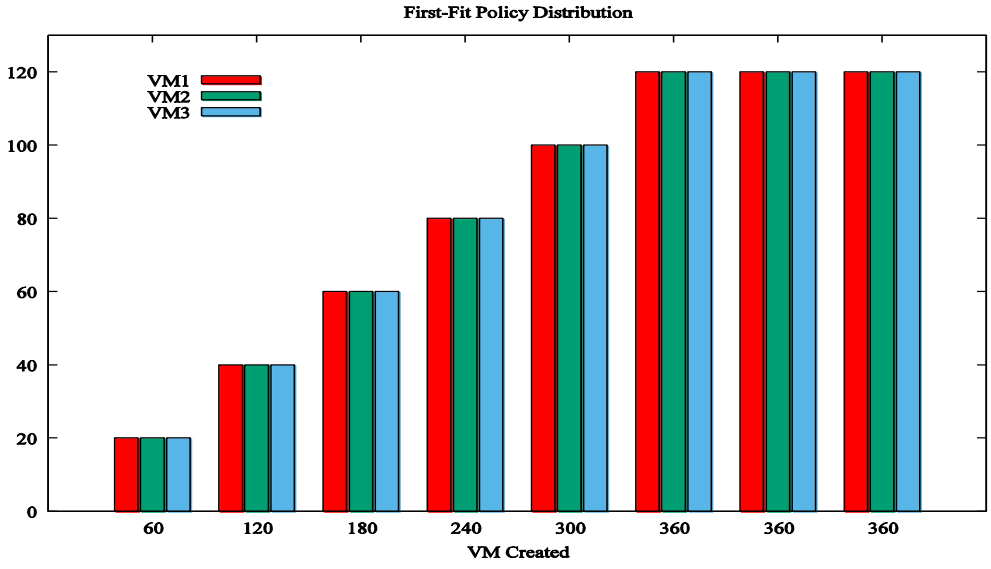

Figure 7. 16: The Distribution of the First-Fit Algorithm to the Created VMs.

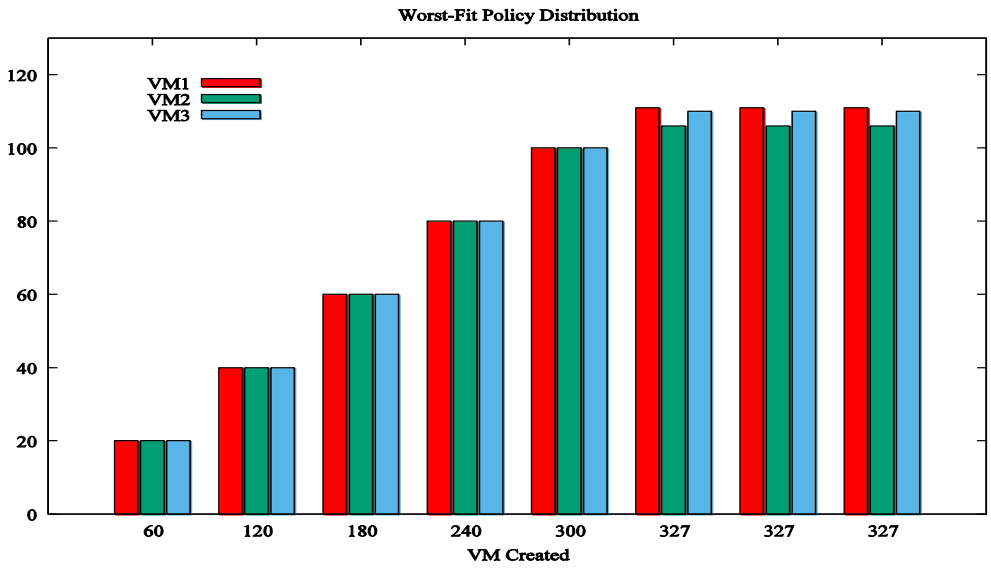

Figure 7. 17: The Distribution of Worst-Fit Algorithm to the Created VMs.

The First-Fit algorithm almost achieved the best turnaround times among Bin-Packing algorithms, especially in over-request cases, due to the good allocation times in over-request cases, VMs creation rate and the balanced distribution of created VMs types c.f. figures 7.13, 7.14 and 7.16. However, the Best-Fit achieved the Worst turnaround time through all request cases among the Bin-Packing algorithms, c.f. figure 7.14.

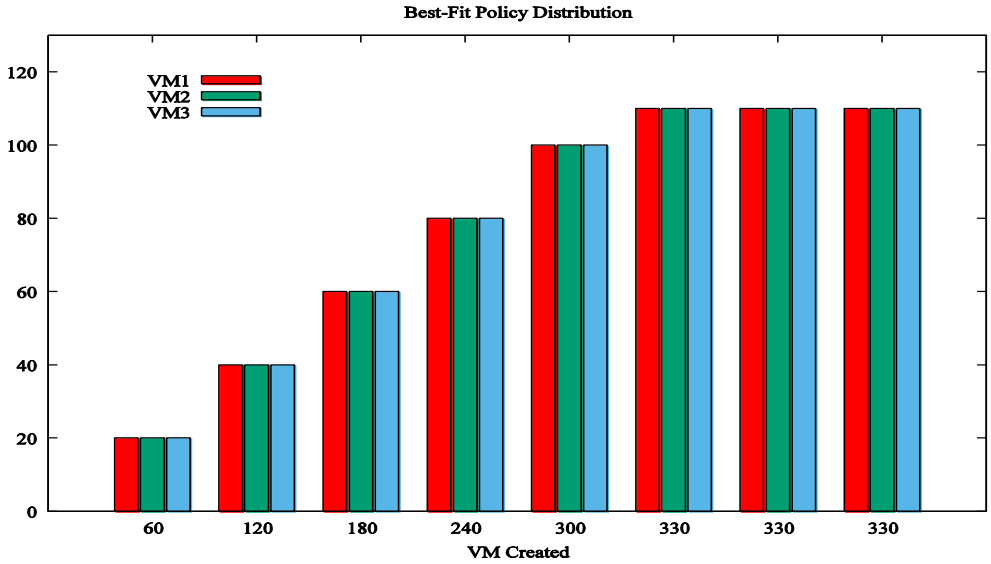

Figure 7. 18: The Distribution of Best-Fit Algorithm to the Created VMs.

According to the Distribution figures (7.15-7.18), there is a gap in the VMs creation rate between: firstly, Agent-based and First-Fit algorithms which created the same amount of VMs at the end. And secondly, the other algorithms that created a different amount of VMs. Agent-based and First-Fit achieved the highest creation rate with well-balanced distribution amongst the VMs types compare with other algorithms, especially the Worst-Fit one.

Note that, the similarity in verification criteria structure between Agent-based (First-Pass-Fit) and First-Fit, helps a First-Fit to achieve the same VMs creation rate and keep the same distance from Agent-based through turnaround time, c.f. figures 7.14-7.16.

Finally, the Agent-based policy has exhibited high efficiency corresponds to other algorithms during the scenario in terms of time including allocation and turnaround times c.f. figures 7.13 and 7.14, and allocation process including the VMs creation rate and VMs types distribution with the First-Fit c.f. figures 7.15-7.16.

7.3.2.2 Scenario II: VMs Diversity vs. Hosts Diversity

The scenario imitates the real-life situation, where in real cloud datacenter there is a variety of hosts' types and the VMs are varying through the requesting process. In terms of that, the scenario has five host types in the datacenter and five VM types are requested through the requesting cases in numerical experiments.

Table 7. 7. The VMs Features.

Feature	*VM 1*	*VM 2*	*VM 3*	*VM 4*	*VM 5*
Image size	10000 MB	10000 MB	10000 MB	10000 MB	10000 MB
RAM	2 GB	2 GB	3 GB	3 GB	4 GB
MIPS	1,500	1,000	2,000	3,000	1,500
Bandwidth	1000	2000	2000	2000	2000
Number of Pes	1	1	1	2	2
VMM	XEN	XEN	XEN	XEN	XEN

Conditions and assumptions of the scenario: First, using five types of VMs as shown in table 7.7. Second, using five types of Hosts as shown in table 7.8, Third, the number of hosts is

fixed (200 hosts); divided into five groups equally, 40 hosts for each type. Forth, the number of requested VMs is varied, but the requesting amount of each VM type is equal.

Table 7. 8. The Hosts Specifications.

Feature	*Host 1*	*Host 2*	*Host 3*	*Host 4*	*Host 5*
Number of Pes	4	5	5	4	3
MIPS	2000	1500	3000	2000	3000
RAM	6 GB	6 GB	8 GB	8 GB	4 GB
Bandwidth	10,000 MHz	10,000 MHz	10,000 MHz	10,000 MHz	10,000 MHz
Storage	1 Terabyte	1 Terabyte	1 Terabyte	1 Terabyte	1 Terabyte

The impact of Hosts and VMs diversity is very obvious over the allocation time (allocation process) on Bin-Packing algorithms, where the differences between all Bin-Packing algorithms appear exactly during all request cases. The First-Fit and Best-Fit during all request cases (under, over) show stable functionality and scored gradually values of allocation time until the end, but the allocation time of Best-Fit is higher than other algorithms through all numerical experiments.

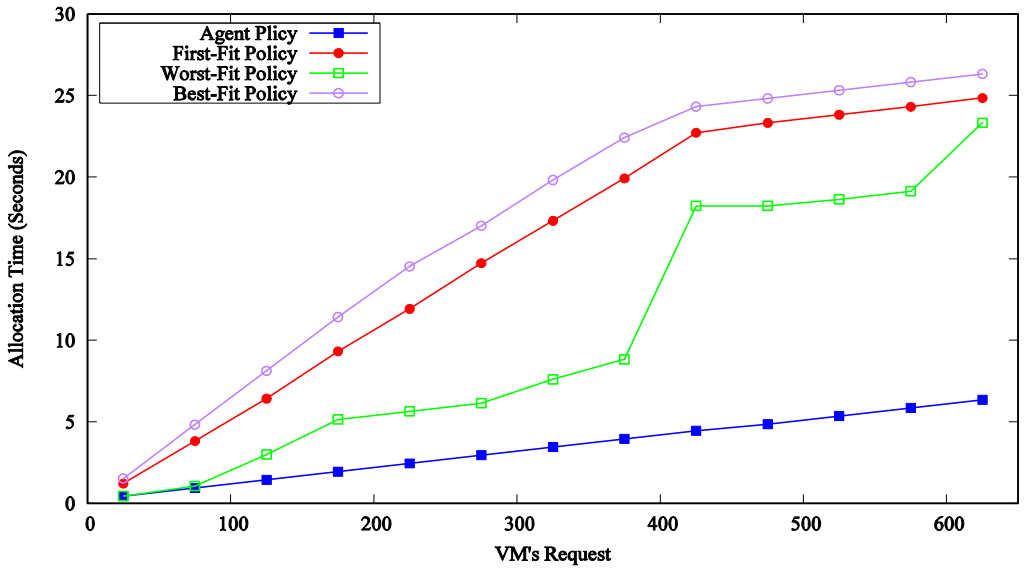

Figure 7. 19: The Allocation Time of Algorithms.

The Best-Fit algorithm in figure 7.19, scores very close values to the First-Fit through all request cases, which makes its curve parallel with the First-Fit curve. Otherwise, the Worst-fit shows unstable performance during all request cases, whereas its allocation time curve takes the zigzag shape. Despite of that, Worst-Fit scores the best allocation time amongst the Bin-Packing algorithms from start until end c.f. figure 7.19.

Agent-based again shows again high efficiency in allocation time over all requesting cases, furthermore, it exposes very steady and stable performance by the smoothly graded values of allocation time during the numerical experiments of this scenario c.f. figure 7.19.

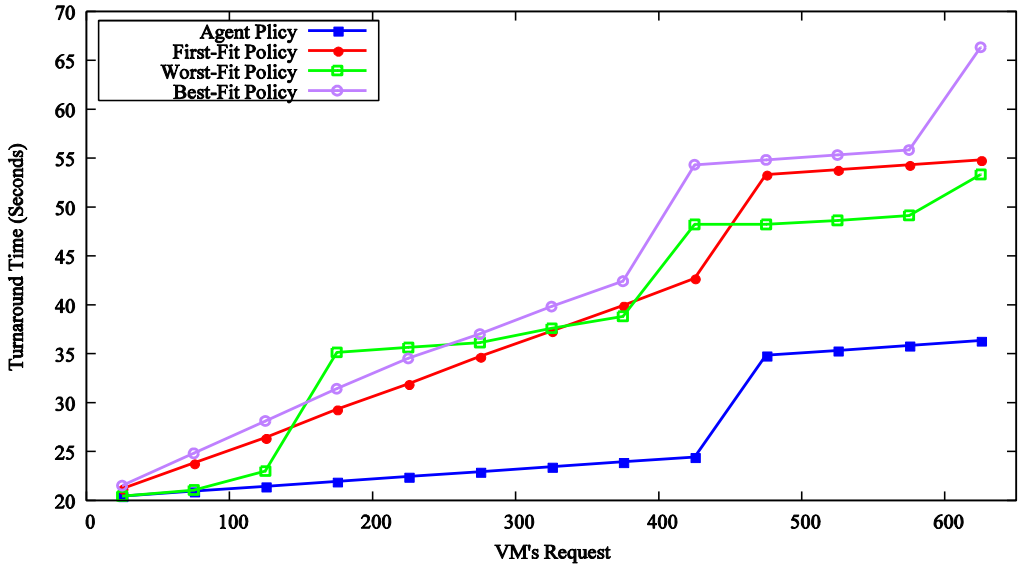

Figure 7. 20: The Turnaround Time of Algorithms.

Consequently, Agent-based algorithm shows high efficiency in the turnaround time through all the requesting cases, further it scores the best values of turnaround time during all numerical experiments of the scenario compare with other algorithms c.f. figure 7.20. The Agent-based has achieved the best values of allocation time c.f. figure 7.19, and it occupied along First-Fit the maximum amount of resources in PEs and MIPS, with a balanced distribution of VMs' type's c.f. figure 7.21, that leads Agent-based to achieve the best values of turnaround time.

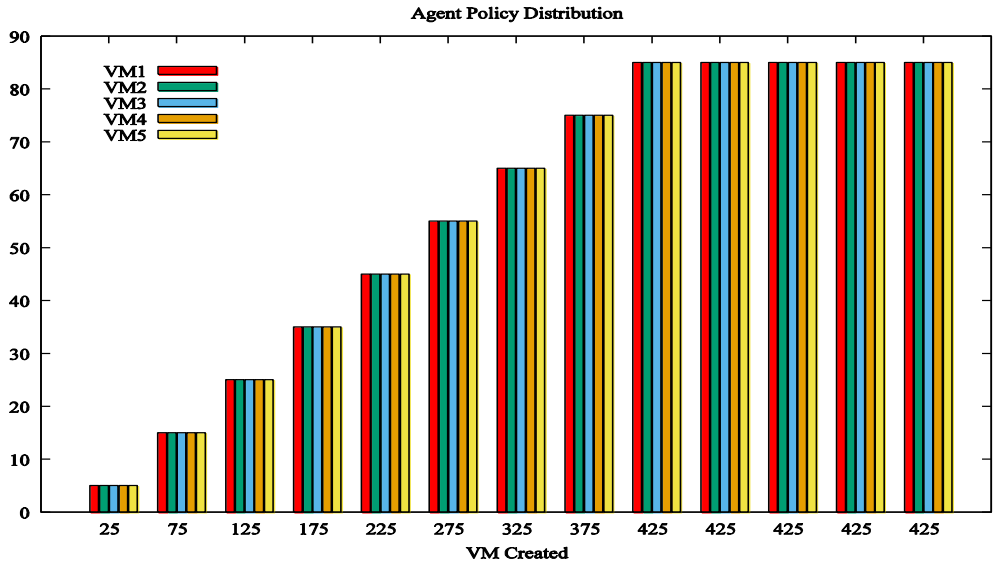

Figure 7. 21: The Distribution of Agent-based Algorithm to the Created VMs.

The Bin-Packing algorithms in turnaround time figure 7.20 follow almost the same manner in allocation time figure 7.19, where the Worst-Fit algorithm takes the leading in the beginning over the Best-Fit and First-Fit even at the end it returns to achieve the best time. Also, Worst-Fit algorithm lost the leading to the First-Fit or to both in some request cases such as 175 VMs and 375 VMs. The First-Fit and Best-Fit show steady and stable performance corresponding to Worst-Fit in turnaround time over all requesting cases and they almost keep the distance with Agent-based fixed, c.f. figure 7.20.

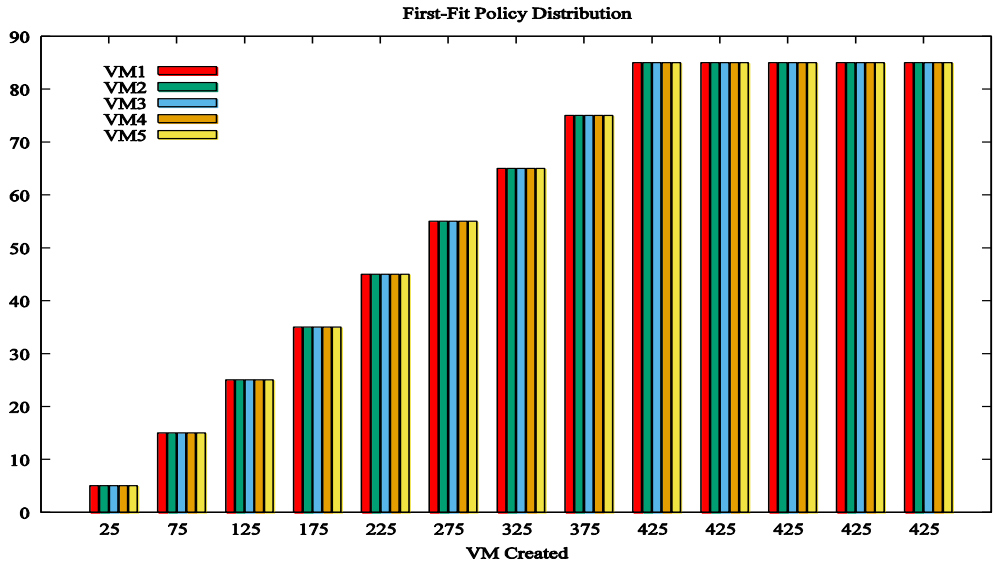

Figure 7. 22: The Distribution of the First-Fit Algorithm to the Created VMs.

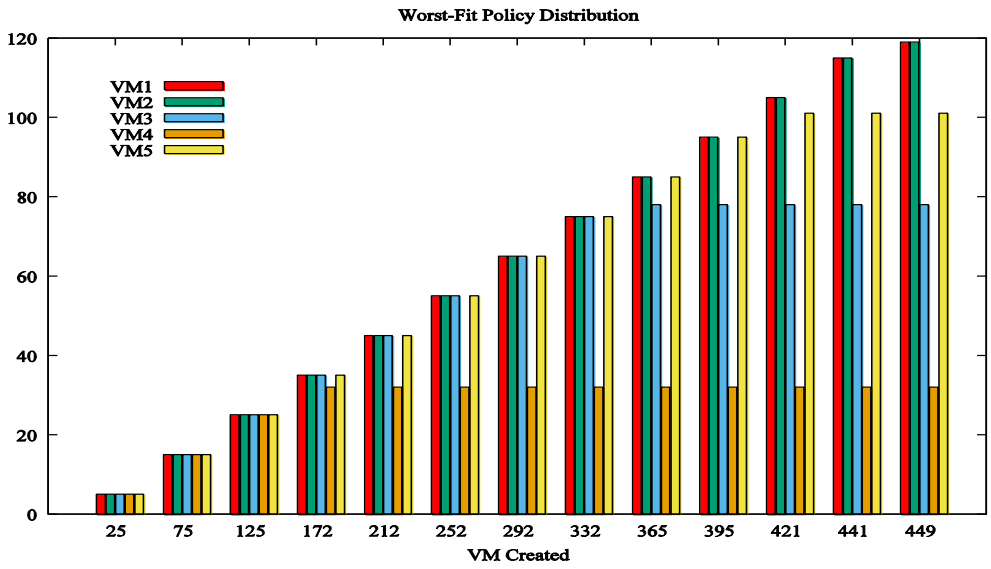

Figure 7. 23: The Distribution of Worst-Fit Algorithm to the Created VMs.

The jumps/leaps in figure 7.20 come from irregular (optimal) distribution of Cloudlets over the created VMs as mentioned before. Agent-based and First-Fit have almost a regular/steady creation process over all request cases with a balanced distribution of VMs' types c.f. figures 7.21 and 7.22, this helps Agent-based to make smoothly/steady turnaround time curve with one leap/jump compare with others.

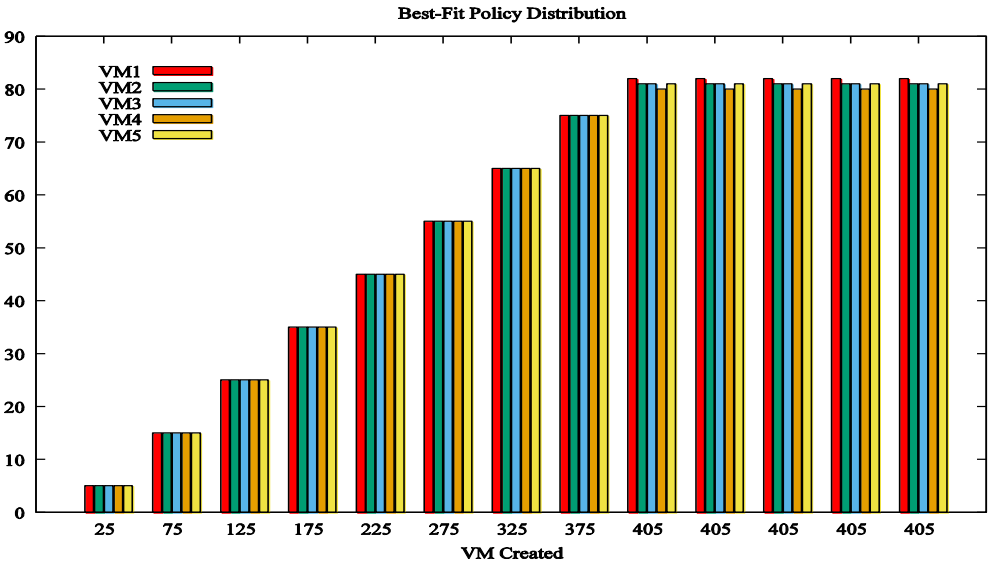

Figure 7. 24: The Distribution of Best-Fit Algorithm to the Created VMs.

According to figures 7.21-7.24, the Worst-Fit algorithm takes the first place of the VM creation rate but the distribution of VMs' types is an average balance, and the last place of the VM creation rate went to Best-Fit one with good balance distribution of VMs' types. However, The Agent-based and First-Fit algorithm take the second place of VMs creation rate, but they occupied PEs and MIPS more than the Worst-Fit because they give smoothly balance the distribution of VMs' types.

Technically, the hosts and VMs diversity/variety impact is very obvious on Bin-Packing algorithms during all numerical experiments, whereas it shows up the differences between the Bin-Packing algorithms in terms of time (including allocation and turnaround) and creation process (creation rate, distribution), hence the two-dimensional verification criteria makes that very clear compared with last chapter scenarios.

Eventually, the good awareness about all datacenter resources which built by using a multi-agent system, gives the Agent-based algorithm the leading over all measurement aspects including allocation time, turnaround time, occupied resources (PEs and MIPS) and balanced distribution of VMs' types in this scenario. Also, this gives induction about the Agent-based potential to work in a real cloud datacenter because it shows high efficiency and functionality with complicated scenarios.

After introducing two scenarios through this sub-section, the next section presents one scenario about a heterogeneous datacenter, which includes Time-Shared and Space-Shared hosts simultaneously.

7.4 The Heterogeneous Datacenter

This section tries to imitate a real-life situation through one scenario about the heterogeneous datacenter, where in real cloud datacenter, there are many different resources and provisioning algorithms. Therefore, the current datacenter has heterogeneous pooling resources (hosts), this means the datacenter has a combination of Time-Shared and Space-Shared hosts simultaneously.

Moreover, in this scenario, all Bin-Packing algorithms in the last two sections are compared with Agent-based, which means the six algorithms have two-dimensional verification criteria

and are divided into two groups: (i) three belong to Time-Shared, and (ii) other three belong to Space-Shared.

Additionally, this section tries to verify/test the dynamic/elasticity of algorithms among different resources provisioning algorithms such as the host level schedulers (Time-Shared, Space-Shared); so that the impacts of PEs and MIPS are considered (two-dimensional) just only. Further, the impacts of other dimensional such as RAM, BW and Storage are avoided, where the datacenter hosts have the penalty amount from those resources (dimensional).

This scenario is similar to Sub-Section 6.4.2, so the requesting cases are verified by using a fixed number of hosts in the datacenter, also it aims to verify the functionality of all algorithms over the heterogeneous resources pooling (datacenter) from under-requested cases up to the over-requested cases.

The conditions and assumptions: First, Using one type of VM. Second, each VM needs: one PE, 1,000 MIPS, 1 GB of RAM and 1000 for Bandwidth. Third, using two types of Hosts; which are divided equally (50% for each type). Forth, the both host types have same specifications/features except the schedulers, where first half uses Time-Shared and second one uses Space-Shared. Fifth, the number of requested VMs is variety. Sixth, the number of hosts (datacenter capacity) is fixed (100 hosts).

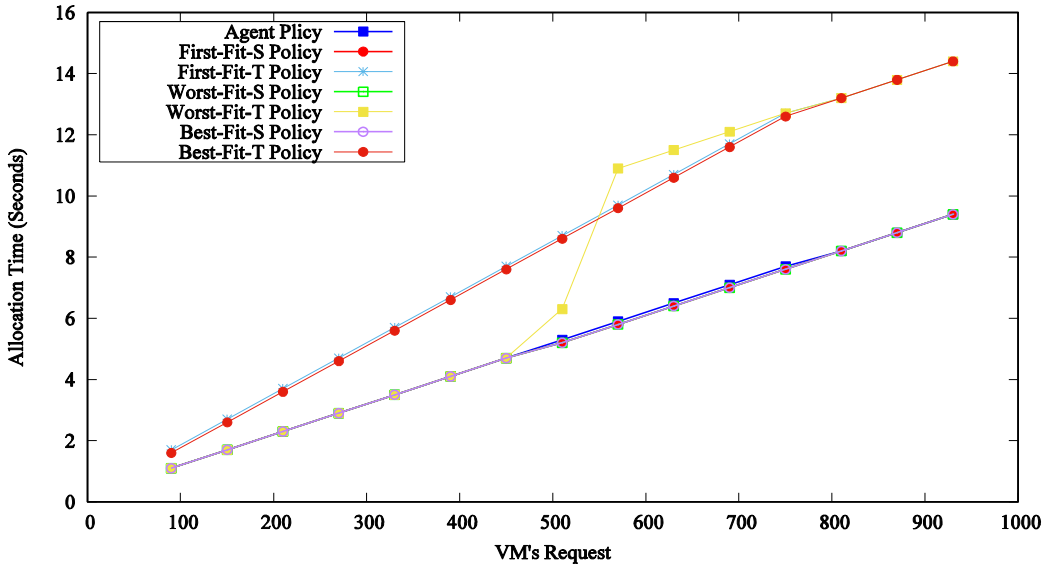

Figure 7. 25: The Allocation Time of all Algorithms.

The configuration of scenario is not complicated because it just uses one VM type and the hosts have plenty of resources corresponds to the VM's specifications (requirements). According to that, the requesting-cases can be verified in smoothly way and gradually to test the efficiency and dynamic of all algorithms through these cases.

According to figure 7.25, all Space-Shared Bin-Packing algorithms and Agent-based achieved at the end the best allocation time compare with all Time-Shared Bin-Packing algorithms. Due to the Space-Shared scheduler locks/reserves the host PE for just one VM; this means reducing the number of creation attempts which leading to reduce allocation time. Further, the gap between the first group and second was very small at the beginning, but it became bigger at the end.

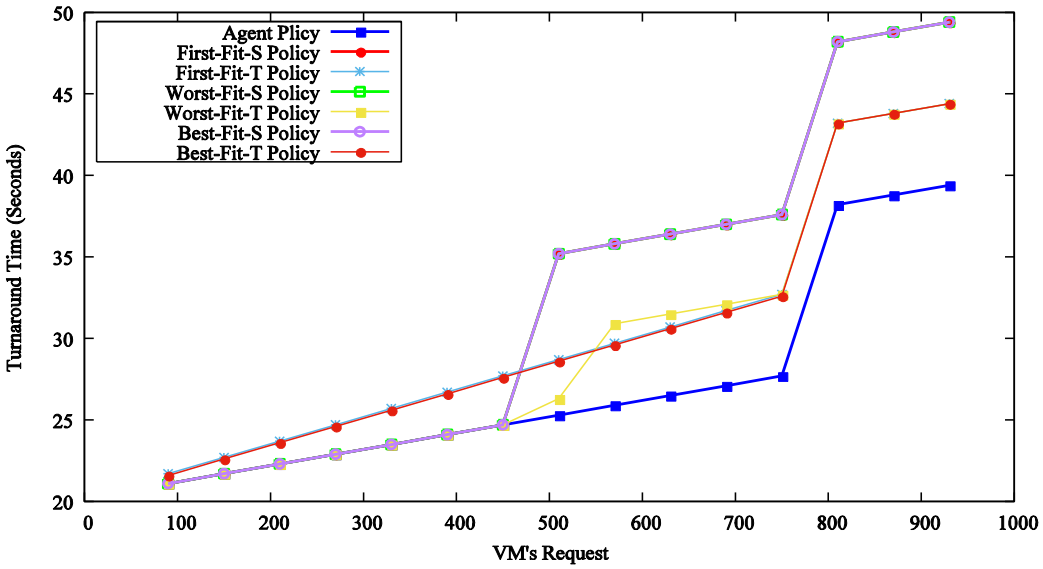

Figure 7. 26: The Turnaround Time of all Algorithms.

In turnaround time figure 7.26, algorithms at end divided into three groups: (i) Agent-based algorithm which achieved the best turnaround time over all other algorithms. (ii) Time-Shared Bin-Packing algorithms which took the second place. (iii) Finally, Space-Shared algorithms which come in the last place. Due to the VMs creation rate, the Time-Shared algorithms exceeded the Space-Shared algorithms in turnaround time c.f. figure 7.26, where

Time-Shared Bin-Packing algorithms and Agent-based created VMs (750 VMs) more than all Space-Shared algorithms (500 VMs).

Aforementioned, the Space-Shared algorithms have performed perfectly with just one-half of the datacenter (Space-Shared hosts), where they allocated/created the maximum allowed rate of VMs by that half with short allocation time. But, in the other half (Time-Shared hosts) they achieved the minimum allowed rate of VMs (fewer creation attempts); this makes them score good allocation time. But, that leads to waste available resources and achieve worst turnaround time c.f. the turnaround time in Section 5.1.

In the same context, Time-Shared algorithms follow the same manner of Space-Shared ones, where they perform perfectly with Time-Shared hosts but badly with other hosts. According to that, they achieved bad allocation time and good turnaround time compared with Space-Shared algorithms.

However, Agent-based exhibited high dynamic/flexible manner over allocation time and turnaround time compared with other algorithms. Due to, firstly it covers all VM specifications (full-dimensional) through the verification process, and secondly the high dynamic/flexible structure because it depends on the host itself (host provisioning policies/algorithms) to do the verification process, updating process and creation process. Furthermore, the using of the multi-agent system helps the Agent-based policy to do the previous two points and perform perfectly with two halves of the datacenter (Space-Shared, Time-Shared) simultaneously.

Note that, the CloudSim toolkit has just one provisioning policy/algorithm to provision RAM and BW in hosts through the VM allocation process. So, if new policies/algorithms are proposed in future for RAM provisioning like MIPS, that means developing more versions/copies for conventional allocation algorithms such as Bin-Packing ones to be compatible with RAM provisioning policies for example. While in Agent-based will not need any developing; because it depends on the host resources provisioning policies/algorithms like RAM and BW as mentioned before.

Practically, Agent-based algorithm showed high dynamic/flexible performance through different provisioning policies, along with the efficiency through the allocation process compared with the conventional allocation algorithms such as Bin-Packing algorithms.

The next section will introduce a brief discussion to all chapter sections.

7.5 Discussion

In this chapter, three Bin-Packing algorithms were configured with Time-Shared and Space-Shared schedulers, creating in this way six policies. These policies were compared against the proposed Agent-based policy over five different scenarios. The latter were divided into three categories Time-Shared, Space-Shared and heterogeneous (Time & Space Shared) datacenter in a host scheduling level based on CloudSim toolkit standard. Moreover, two measurement criteria were adopted to evaluate the results of all algorithms, which are allocation and turnaround times and allocation process (VMs creation/resources occupied).

Additionally, two-dimensional designs (number of PEs and MIPS) were adopted to Bin-Packing algorithms for verifying hosts in a datacenter during numerical experiments associated with the scenarios. According to that, each Bin-Packing algorithm had two versions/copies during the numerical experiments of this chapter; because the CloudSim toolkit had two schedulers for the PEs and MIPS (Time-Shared, Space-Shared). On the other hand, the Agent-based policy kept the same design, which was used in Chapters 5 and 6 without any modification or improvement.

The employment of the multi-agent system showed that the Agent-based policy was superior over all other used algorithms during all scenarios in terms of time and VM allocation/placement process. In this context, the Agent-based policy clearly achieved the best allocation time and turnaround time among all scenarios in the three defined categories as compared with the other algorithms of the study. Furthermore, the Agent-based exhibited high performance in VM allocation/placement process, where it created in most scenarios with the highest VMs creation rate (which refers to the total number of created VMs during the numerical experiments or inter the scenario) with a balanced distribution for VMs types such as those of Sub-Sections7.2.2.1 and 7.3.2.1. On the contrary, in other scenarios, such as those of Sub-Sections 77.2.2.2 and 7.3.2.2, other algorithms created a greater number of VMs in comparison to the number of VMs of the Agent-based. However, the Agent-based policy and First-Fit algorithms achieved the highest amount of occupied resources especially in computational power (PEs, MIPS) with a balanced distribution for VMs types.

Due to the using of a multi-agent system, the Agent-based algorithm showed a full awareness about the datacenter resources (resources pooling). This facilitated very precise allocation decisions during the allocation process in order to select a proper host to requested VM and showed high dynamic/flexible performance over the schedulers. Consequently, Agent-based achieved most efficient results in terms of time and allocation process compared with conventional algorithms, especially in a heterogeneous datacenter situation.

Finally, the Agent-based policy showed an efficient performance and functionality with high complexity cases/scenarios such as varying VMs' types, varying hosts' types and a heterogeneous datacenter. In other words, the proposed Agent-based policy has a very high potential to work efficiently in a real-life cloud datacenter.

Moreover, the Agent-based policy exhibited superior flexibility amongst the heterogeneous cloud datacenter, where it performs most efficiently and steadily under both schedulers, namely Time-Shared and Space-Shared.

7.6 Summary

This chapter presented another comparative study between the proposed Agent-based policy and state of the art algorithms, where three two-dimensional Bin-Packing policies were configured in the CloudSim toolkit in order to compare them against the proposed Agent-based policy. Thus, all policies were verified/tested under both host level schedulers (c.f., Time-Shared, Space-Shared) over five scenarios, two scenarios for each scheduler and the last one for the heterogeneous datacenter (c.f., Time & Space Shared) for covering most cases in a Cloud datacenter.

Next chapter introduces the final conclusions of this thesis.

Chapter 8 Conclusions and Future Work

This chapter presents the main conclusions of this thesis and the future research directions.

8.1 Conclusions

The complexity of the VM allocation/placement process through a datacenter of a cloud computing platform is based on virtualized technology and it is increasingly scalable, subject to the number of hosts/PMs. Due to power consumption, QoS, resources utilization and high response. To this end, the importance of the design and development of new algorithms/policies towards a VM allocation/placement has attracted considerable research interest in the literature. In this thesis, a quantitative study was proposed, towards the design, development and validation of a new Agent-based VM allocation/placement policy through the virtualized datacenters of cloud computing (c.f., Chapter 4) This policy was experimentally assessed and favorable comparisons were made against default algorithm adopted by CloudSim toolkit and 4 algorithms of the stat of the art under different scenarios (c.f., Chapters 5-7). ,

Most of the current VM allocation policies and consolidation and optimization approaches, which are used during the VM placement progress (find/identify the host/PM), are conventional algorithms or mechanisms, such as the Bin-Packing algorithms, based on a modified version of the BFD (Best-Fit Decreasing) algorithm [7-12]. Consequently, these algorithms discard the influence of time consumption during find/identify the proper host/PM to the requested VM and the VM creation operation.

In contrast, the proposed Agent-based policy uses the multi-agent system technology that makes the VM allocation/placement decision very accurate and precise. In this way, the time consumption for the allocation is reduced through the placement process, which includes the finding/identifying of the proper host and VM creation stages. Specifically, by using the contract net protocol of the multi-agent system (c.f. Section 4.2) through the VM allocation process amongst

the resources of the datacenter, it makes the allocation/placement process more efficient and also avoids failed VM creation attempts.

As this thesis focuses on the allocation process phase through the VM lifecycle in a cloud computing datacenter, it carries out an analysis of the mechanism and techniques of VM allocation/placement algorithms, such as the bin-packing algorithms. In this context, the Agent-based policy is proposed as a credible solution, which mitigates the impact of the complexity amongst the virtualized mega-datacenter and minimizes the allocation time. To this end, this thesis employs as measurements the allocation time and amount of occupied resources, and standards in order to evaluate the proposed new Agent-based policy and other algorithms during the numerical experiments of different scenarios.

The CloudSim toolkit with its novel features was selected to simulate the cloud computing platform, and evaluate the algorithms against the proposed Agent-based policy. Specifically, the CloudSim toolkit was very helpful due to its special features for the design and development of new provisioning or scheduling algorithms, such as the VM allocation policies. The breakdown of the CloudSim toolkit classes in Chapter 3 is used to illustrate how to model and implement the cloud computing components like: Broker, Datacenter, Host, and VM…etc., how these components relate and linked and how each component works.

In Chapter 3, three significant topics namely, VM lifecycle (allocation dialog), VM creation and allocation issues and the VM schedulers (Time-Shared [17], Space-Shared [18]) were employed in order to comprehend and evaluate the VM allocation process. In this context, the dialog of the allocation revealed the full story of VM lifecycle, from VM request until to VM destruction and moreover, clarified the role of the VM allocation policy through a VM lifecycle. Moreover, additional important issues to the VM allocation and creation process were introduced, such as VM specifications, host resources, provisioning algorithms and the impact of those issues. In this context, scheduling levels and types were explained through the 'VM schedulers' topic in order to exhibit the real impact of each scheduling type (c.f., time, space) on the functionality of the VM

allocation policies, especially at the host level. Finally, the default VM allocation policy of CloudSim [40] was also presented as it was subsequently used in Chapter 5 to evaluate the concept of using the multi-agent system through the allocation process in a virtualized cloud datacenter.

The proposed new Agent-based VM allocation policy was fully explored in Chapter 4, by focusing on the importance of designing and developing of new VM allocation algorithms/policies, the benefits of using the multi-agent system to the allocation process and the allocation time and occupied resources as new measurement standards. In particular, the new multi-agent system was described in Section 4.3, as the conceptual basis of the new Agent-based VM allocation policy in the cloud datacenter, based on the Partial-Global Planning and Negotiation approaches, as appropriate. Moreover, the communication and coordination systems amongst the system agents, based on the Contract Net Protocol approach [60], were fully illustrated.

Since the design and implementation of the Agent-based policy are compatible and adjustable to the CloudSim toolkit configurations, three Java classes were constructed in order to cover all requirements and specifications of the new allocation policy. The first class for the new Vm_Allocation_Policy_Agent, which is an extension of the abstract class of *Vm_Allocation_Policy* [74];. The second class acts as a Datacenter_Coordinator, which is the responsible of a coordinator-agent object instantiation. And the last one is a class of Host_Agent, which is in charge of interleaf-agent object instantiation. Technically, the multi-agent system through this thesis was composed of two simple Make-Decision agent types (including Datacenter_Coordinator and Host_Agent), that consists of just one Agent-Class (group or flock) distributed on two layers.

According to the design and implementation of the multi-agent system in Chapter 4, the CloudSim creates just one object of Vm_Allocation_Policy_Agent and Datacenter_Coordinator for each datacenter and one object of Host-Agent for each host/PM in the datacenter during all numerical experiments in Chapters 5-7. Further, the Contract Net Protocol is used between the Datacenter_Coordinator and Host_Agent objects through the allocation process in order to identify and find the proper host/PM to the requested VM. The new Agent-based policy was undertaken its own special testing and selection criteria based on

the Contract Net Protocol, which consists of two stages, namely (i) On the host-level, where Host_Agent object uses the member method *Is_Host_Suitable* of the host class [42] in order to verify the ability of the host to host/create the requested VM and return a Boolean result to the Datacenter_Coordinator object (ii) On datacenter-level, where the Datacenter_Coordinator object selects in ascending order (First-Pass-Fit) one of the passed hosts/PMs. In summary, it was shown that the proposed new Agent-based policy employed the multi-agent system, the communication approach of the Contract Net Protocol. And the special testing and selection criteria. Consequently, a good awareness was built about the resources pooling of the datacenter in order to improve the allocation/placement progress amongst the datacenter hosts/PMs.

In Chapter 5, broad comparisons between the proposed new Agent-based policy (c.f., Chapter 4) and default VM allocation policy of CloudSim toolkit (c.f., Chapter 3) were introduced in order to illustrate the concept of using the multi-agent system through the allocation process in cloud datacenters. Therefore, Chapter 5 stipulated the same fixed general conditions and assumptions for the configurations of the numerical experiments during different scenarios. Moreover, a numerical definition (formula) for the evaluation standards to the VM allocation policies such as the allocation time, the cumulative allocation and turnaround times, was introduced. Consequently, the results and analysis of six scenarios were presented in order to assess the potential of the two policies over a wide range of cases and situations. The scenarios were divided into two categories, namely the one based on the Time-Shared Scheduler, and the second one based on the Space-Shared Scheduler. It was experimentally concluded that the new Agent-based policy displayed higher performance and functionality during all scenarios as well as stability amongst the two schedulers of the PEs and MIPS compare with the default one.

Chapter 6 presented a comparative study involving the proposed Agent-based policy versus four other state of the art algorithms, namely the Random algorithm and three One-Dimensional Bin-Packing algorithms (First-Fit, Worst-Fit and Best-Fit). In this context, four extended Java classes from *Vm_Allocation_Policy* [74] abstract class were implemented, for instantiating the state of the art algorithms through the CloudSim toolkit environment. However, the evaluation standards and measurements, the general conditions and assumptions of the numerical experiments and the Agent-based policy in Chapter 6 were

similar to those of Chapter 5 without any modification. Moreover, the verifications and comparisons performed through six scenarios were divided into two groups as in Chapter 5 as follows: the first group uses the Time-Shared scheduler [17] and the second one uses Space-Shared scheduler [18]. Note that the one-dimensional structure of the three Bin-Packing algorithms was based on the number of PEs in the host/PM whilst the complexity of them is quadratic of order O (n^2). According to the results and analysis of the six scenarios under the high complexity situations with both schedulers, the proposed new Agent-based policy, based on the multi-agent system, showed again superior performance and stability as compared with the other algorithms over the metrics consisting of the Allocation time, Turnaround time and amount of occupied resources.

Finally, in Chapter 7, the three Bin-Packing algorithms (c.f., Chapter 6) are implemented based on two-dimensional verification/testing criteria, namely i) the number of PEs in the host/PM and ii) the MIPS of each PE. By employing the Time-Shared and Space-Shared schedulers on the host-level for the PEs and MIPS in conjunction with the static/fixed manner of the conventional algorithms, two versions for each Bin-Packing algorithm were implemented. To this end, six extended Java classes from *Vm_Allocation_Policy* [74] abstract class are implemented, using three policies for the Time-Shared scheduler and another three policies for the Space-Shared scheduler. However, the Agent-based policy and the six policies were verified through four scenarios with moderate complexity, which are divided equally between the two schedulers, two scenarios for each one. Also, the structure and implementation of the proposed new Agent-based policy, the evaluation standards and measurements and the general conditions and assumptions of the numerical experiments of Chapter 7 were identical to Chapter 5.

Moreover, Chapter 7 introduced an additional special scenario, which aims to verify/test the dynamic characteristic of each of the seven considered policies amongst different scheduling or provisioning algorithms of the host/PM. Thus, this scenario imitates a real-life datacenter situation, which is a heterogeneous configuration (including different scheduling and provisioning policies) of the pooling-resource of the datacenter. According to that, the Time-shared, Space-Shared schedulers were used simultaneously, where 50% of hosts/PMs used Time-Shared scheduler and the rest 50% used the Space-Shared scheduler. Moreover, as the main objective of this scenario is checking the dynamic characteristic of policies, during the

numerical experiments just the impacts of PEs and MIPS are considered whilst the impacts of other measures, such as RAM and BW (Band-Width) are not taken into account. Based on the results and observations of all these scenarios, the Agent-based policy exhibited again a better performance in terms of time and allocation process over all other algorithms under consideration. This performance superiority was particularly noticeable in more complex situations and cases, also the new Agent-based policy showed a high dynamic performance (flexibility) over the different scheduling and provisioning policies such as Time-Shared and Space-Shared through the heterogeneous cases (c.f., Section 7.4).

8.2 Recommendations for Future Work

Possible extensions of the work include the following research directions and associated applications in Cloud computing and virtualized datacenter:

- Verify the current Agent-based VM allocation policy structure, implementation and objectives with more flexible factors. Due to, the general factors during all numerical experiments of this work were fixed; using flexible and dynamic factors will open new research directions and challenge, like the following factors:
  - Datacenter: verify the proposed new Agent-based through cloud computing environment with multiple datacenters, like current real cloud computing (Amazon, Google, Microsoft….), where most of the commercial cloud computing now includes more than one datacenter.
  - Broker: using multiple Brokers through the verification process; because in real scenario/situation the cloud computing deals with queries (VMs) of many Brokers simultaneously.
  - Cloudlet: using different Cloudlets/applications with different specifications, which need to different VMs types according to the Cloudlets requirements.
- Change the selection criteria of the Agent-based policy, where the current one is First-Pass-Fit which depends on the concept of the Bin-Packing First-Fit algorithm. Hence, there is potential to use another concept of Bin-Packing algorithms, such as a Best-Fit, Decreasing Best-Fit …etc.; which might improve the performance and functionality of the Agent-based policy.

- Reconfigure and modify the current Agent-based policy structure and implementation, to serve special applications, such as HPC (High-Performance Computing), Social Networking and BD (Big Data) processing.
- Use the new Agent-based policy in order to new purposes and objectives, because the Agent-based policy in this thesis focused on the VM allocation/placement process just only. This means the current multi-agent system, which used through the Agent-based policy needs to some development and modification for serving the new objectives, such as the optimization of the VM allocation process. However, most of the modification will focus on the agent structure and implementation especially the interleaf agent (Host_Agent); to improve the verification criteria and make the agent more intelligent according to the new objectives and goals, like the following optimization purposes:
    - Power-Consumption: it is a hot research topic now in the cloud computing field, where most of the optimization work for the VM allocation, aims to propose new approaches and methodologies to reduce the power consumption in the cloud datacenters depends on the VM live-migration concept.
    - QoS (Quality of Service): which is usually defined based on the SLA (Service Level Agreement), must be reserved or protected during any optimization progress, such as power consumption, by using prevent QoS violations approaches.

  However, Chapter 2 (the Related Works) included some approaches and policies to optimize the VM allocation according to many reasons and objectives, those objectives can be taken into consideration through the future developing of the multi-agent system of the new Agent-based policy.

- Finally, extend the multi-agent system of the prosed new Agent-based policy; because the multi-agent system in this thesis context is composed of just one Agent-Class, which is distributed over just two layers. Thus, there is a possibility to extend the multi-agent system to be composed of multiple Agent-Class and distributed over multi-layer (more than two layers); to verify and evaluate new agent mechanisms and protocols for communicating and coordinating to perform the tasks and jobs like, VM allocation

and optimization. Moreover, this direction is significant to apply the new Agent-based policy through the N-Tier datacenter design style, which is used to the model of the N-Tier application.

References

[1] T. Dillon, C. Wu, and E. Chang, "Cloud computing: issues and challenges," in *Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on*, 2010, pp. 27-33.

[2] I. Foster, Y. Zhao, I. Raicu, and S. Lu, "Cloud computing and grid computing 360-degree compared," in *Grid Computing Environments Workshop, 2008. GCE'08*, 2008, pp. 1-10.

[3] L. Youseff, M. Butrico, and D. Da Silva, "Toward a unified ontology of cloud computing," in *Grid Computing Environments Workshop, 2008. GCE'08*, 2008, pp. 1-10.

[4] R. Buyya, R. Ranjan, and R. N. Calheiros, "Modeling and simulation of scalable Cloud computing environments and the CloudSim toolkit: Challenges and opportunities," in *High Performance Computing & Simulation, 2009. HPCS'09. International Conference on*, 2009, pp. 1-11.

[5] W. Hao, I.-L. Yen, and B. Thuraisingham, "Dynamic service and data migration in the clouds," in *Computer Software and Applications Conference, 2009. COMPSAC'09. 33rd Annual IEEE International*, 2009, pp. 134-139.

[6] J. T. Piao and J. Yan, "A network-aware virtual machine placement and migration approach in cloud computing," in *Grid and Cooperative Computing (GCC), 2010 9th International Conference on*, 2010, pp. 87-92.

[7] A. Beloglazov and R. Buyya, "Energy efficient allocation of virtual machines in cloud data centers," in *Cluster, Cloud and Grid Computing (CCGrid), 2010 10th IEEE/ACM International Conference on*, 2010, pp. 577-578.

[8] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," *Future generation computer systems,* vol. 28, pp. 755-768, 2012.

[9] R. Buyya, A. Beloglazov, and J. Abawajy, "Energy-efficient management of data center resources for cloud computing: A vision, architectural elements, and open challenges," *arXiv preprint arXiv:1006.0308,* 2010.

[10] A. Beloglazov and R. Buyya, "Energy efficient resource management in virtualized cloud data centers," in *Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, 2010, pp. 826-831.

[11] A. Beloglazov and R. Buyya, "Adaptive threshold-based approach for energy-efficient consolidation of virtual machines in cloud data centers," in *Proceedings of the 8th International Workshop on Middleware for Grids, Clouds and e-Science*, 2010, p. 4.

[12] A. Beloglazov and R. Buyya, "Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers," *Concurrency and Computation: Practice and Experience,* vol. 24, pp. 1397-1420, 2012.

[13] B. Rieck, "Basic Analysis of Bin-Packing Heuristics," *Publicado por Interdisciplinary Center for Scientific Computing. Heildelberg University,* 2010.

[14] D. Talia, "Cloud Computing and Software Agents: Towards Cloud Intelligent Services," in *WOA*, 2011, pp. 2-6.

[15] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya, "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and Experience,* vol. 41, pp. 23-50, 2011.

[16] R. N. Calheiros, R. Ranjan, C. A. De Rose, and R. Buyya, "Cloudsim: A novel framework for modeling and simulation of cloud computing infrastructures and services," *arXiv preprint arXiv:0903.2525,* 2009.

[17] U. o. M. The Cloud Computing and Distributed Systems (CLOUDS) Laboratory. *Class Vm Scheduler Time Shared.* Available: http://www.cloudbus.org/cloudsim/doc/api/org/cloudbus/cloudsim/VmSchedulerTimeShared.html

[18] U. o. M. The Cloud Computing and Distributed Systems (CLOUDS) Laboratory. *Class Vm Scheduler Space Shared.* Available: http://www.cloudbus.org/cloudsim/doc/api/org/cloudbus/cloudsim/VmSchedulerSpaceShared.html

[19] A. Al-Ou'n, M. Kiran, and D. D. Kouvatsos, "Using Agent-Based VM Placement Policy," in *Future Internet of Things and Cloud (FiCloud), 2015 3rd International Conference on*, 2015, pp. 272-281.

[20] P. Mell and T. Grance, "The NIST definition of cloud computing," 2011.

[21] Google. (2016, July 13). *Google Datacenters*. Available: https://www.google.com/about/datacenters/

[22] Microsoft. (2016, July 13). *Microsoft global datacenters*. Available: https://www.microsoft.com/en-us/cloud-platform/global-datacenters

[23] J. Xu and J. A. Fortes, "Multi-objective virtual machine placement in virtualized data center environments," in *Green Computing and Communications (GreenCom), 2010 IEEE/ACM Int'l Conference on & Int'l Conference on Cyber, Physical and Social Computing (CPSCom)*, 2010, pp. 179-188.

[24] T. Kaur and I. Chana, "Power-Aware Virtual Machine Scheduling-policy for Virtualized Heterogeneous Multicore Systems," 2013.

[25] M.-H. Tsai, J. Chou, and J. Chen, "Prevent VM Migration in Virtualized Clusters via Deadline Driven Placement Policy," in *Cloud Computing Technology and Science (CloudCom), 2013 IEEE 5th International Conference on*, 2013, pp. 599-606.

[26] Google. (2016, Feb. 16). *Google App Engine*. Available: www.appengine.google.com

[27] Amazon. (Feb. 16). *Amazon Elastic Compute Cloud (EC2)*. Available: www.aws.amazon.com/ec2/

[28] Amazon. (Feb. 16). *Amazon Web Services (AWS) - Cloud Computing Services*. Available: www.aws.amazon.com

[29] B. Wilder, *Cloud architecture patterns: using microsoft azure*: " O'Reilly Media, Inc.", 2012.

[30] Microsoft. (Feb. 16). *Microsoft Azure: Cloud Computing Platform & Services*. Available: www.azure.microsoft.com

[31] A. Quiroz, H. Kim, M. Parashar, N. Gnanasambandam, and N. Sharma, "Towards autonomic workload provisioning for enterprise grids and clouds," in *Grid Computing, 2009 10th IEEE/ACM International Conference on*, 2009, pp. 50-57.

[32] R. Buyya and M. Murshed, "Gridsim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing," *Concurrency and computation: practice and experience,* vol. 14, pp. 1175-1220, 2002.

[33] A. Legrand, L. Marchal, and H. Casanova, "Scheduling distributed applications: the simgrid simulation framework," in *Cluster Computing and the Grid, 2003. Proceedings. CCGrid 2003. 3rd IEEE/ACM International Symposium on*, 2003, pp. 138-145.

[34] C. L. Dumitrescu and I. Foster, "GangSim: a simulator for grid scheduling studies," in *Cluster Computing and the Grid, 2005. CCGrid 2005. IEEE International Symposium on*, 2005, pp. 1151-1158.

[35] CloudSim: A Framework For Modeling And Simulation Of Cloud Computing Infrastructures And Services. *CloudSim Project Google Group*. Available: https://code.google.com/p/cloudsim/
[36] U. o. M. The Cloud Computing and Distributed Systems (CLOUDS) Laboratory. *CloudSim ToolKit*. Available: http://www.cloudbus.org/cloudsim/
[37] U. o. M. The Cloud Computing and Distributed Systems (CLOUDS) Laboratory. *CloudSim API*. Available: www.cloudbus.org/cloudsim/doc/api
[38] CloudSim.org. (Feb. 20). *DatacenterBroker Class*. Available: http://www.cloudbus.org/cloudsim/doc/api/org/cloudbus/cloudsim/DatacenterBroker.html
[39] CloudSim.org. (Feb. 20). *Datacenter Class*. Available: http://www.cloudbus.org/cloudsim/doc/api/org/cloudbus/cloudsim/Datacenter.html
[40] U. o. M. The Cloud Computing and Distributed Systems (CLOUDS) Laboratory. *Class Vm Allocation Policy Simple*. Available: http://www.cloudbus.org/cloudsim/doc/api/org/cloudbus/cloudsim/VmAllocationPolicySimple.html
[41] CloudSim.org. (Feb. 20). *DatacenterCharacteristics Class*. Available: http://www.cloudbus.org/cloudsim/doc/api/org/cloudbus/cloudsim/DatacenterCharacteristics.html
[42] CloudSim.org. (Feb. 20). *Host Class*. Available: http://www.cloudbus.org/cloudsim/doc/api/org/cloudbus/cloudsim/Host.html
[43] CloudSim.org. (Feb. 20). *VM Class*. Available: http://www.cloudbus.org/cloudsim/doc/api/org/cloudbus/cloudsim/Vm.html
[44] CloudSim.org. (Feb. 20). *CloudletSchedulerTimeShared Class*. Available: http://www.cloudbus.org/cloudsim/doc/api/org/cloudbus/cloudsim/CloudletSchedulerTimeShared.html
[45] CloudSim.org. (Feb. 20). *CloudletSchedulerSpaceShared Class*. Available: http://www.cloudbus.org/cloudsim/doc/api/org/cloudbus/cloudsim/CloudletSchedulerSpaceShared.html
[46] CloudSim.org. (Feb. 24). *VM Scheduler Class*. Available: http://www.cloudbus.org/cloudsim/doc/api/org/cloudbus/cloudsim/VmScheduler.html
[47] CloudSim.org. (Feb. 24). *Pe Provisioner Class*. Available: http://www.cloudbus.org/cloudsim/doc/api/org/cloudbus/cloudsim/provisioners/PeProvisioner.html
[48] CloudSim.org. (Feb. 24). *RAM Provisioner Class*. Available: http://www.cloudbus.org/cloudsim/doc/api/org/cloudbus/cloudsim/provisioners/RamProvisioner.html
[49] CloudSim.org. (Feb. 24). *BW Provisioner Class*. Available: http://www.cloudbus.org/cloudsim/doc/api/org/cloudbus/cloudsim/provisioners/BwProvisioner.html
[50] CloudSim.org. (Feb. 24). *Pe Provisioner Simple Class*. Available: http://www.cloudbus.org/cloudsim/doc/api/org/cloudbus/cloudsim/provisioners/PeProvisionerSimple.html
[51] CloudSim.org. (Feb. 24). *RAM Provisioner Simple Class*. Available: http://www.cloudbus.org/cloudsim/doc/api/org/cloudbus/cloudsim/provisioners/RamProvisionerSimple.html

[52] CloudSim.org. (Feb. 24). *BW Provisioner Simple Class*. Available: http://www.cloudbus.org/cloudsim/doc/api/org/cloudbus/cloudsim/provisioners/BwProvisionerSimple.html
[53] CloudSim.org. (Mar. 01). *CloudletScheduler Class*. Available: http://www.cloudbus.org/cloudsim/doc/api/org/cloudbus/cloudsim/CloudletScheduler.html
[54] J. R. Searle, *Speech acts: An essay in the philosophy of language* vol. 626: Cambridge university press, 1969.
[55] F. L. Bellifemine, G. Caire, and D. Greenwood, *Developing multi-agent systems with JADE* vol. 7: Wiley. com, 2007.
[56] J. Mayfield, Y. Labrou, and T. Finin, "Evaluation of KQML as an agent communication language," in *International Workshop on Agent Theories, Architectures, and Languages*, 1995, pp. 347-360.
[57] M. Wooldridge and M. Tambe, *Intelligent Agents II-Agent Theories, Architectures, and Languages: IJCAI'95-ATAL Workshop, Montreal, Canada, August 19-20, 1995 Proceedings* vol. 2: Springer Science & Business Media, 1996.
[58] N. R. Jennings, "Commitments and conventions: The foundation of coordination in multi-agent systems," *The knowledge engineering review,* vol. 8, pp. 223-250, 1993.
[59] E. H. Durfee, "Practically coordinating," *AI Magazine,* vol. 20, p. 99, 1999.
[60] R. G. Smith and R. Davis, "Frameworks for cooperation in distributed problem solving," *IEEE Transactions on systems, man, and cybernetics,* vol. 11, pp. 61-70, 1981.
[61] M. P. Georgeff, S. I. M. P. CA., S. I. C. Science, and T. Division, *Communication and Interaction in Multi-agent Planning*: Artificial Intelligence Center, SRI International, 1983.
[62] E. H. Durfee and V. R. Lesser, "Using partial global plans to coordinate distributed problem solvers," *Readings in distributed artificial intelligence,* pp. 285-293, 1988.
[63] E. H. Durfee, V. R. Lesser, and D. D. Corkill, "Coherent cooperation among communicating problem solvers," *IEEE Transactions on Computers,* vol. 100, pp. 1275-1291, 1987.
[64] E. H. Durfee and V. R. Lesser, "Planning coordinated actions in dynamic domains," 1987.
[65] E. H. Durfee and V. R. Lesser, "Predictability Versus Responsiveness: Coordinating Problem Solvers in Dynamic Domains," in *AAAI*, 1988, pp. 66-71.
[66] S. Bussmann and J. Muller, "A negotiation framework for co-operating agents," *Proceedings of CKBS-SIG,* pp. 1-17, 1992.
[67] R. G. Smith and R. Davis, "Frameworks for cooperation in distributed problem solving," *Systems, Man and Cybernetics, IEEE Transactions on,* vol. 11, pp. 61-70, 1981.
[68] R. T. Chi and E. Turban, "Distributed intelligent executive information systems," *Decision Support Systems,* vol. 14, pp. 117-130, 1995.
[69] N. K. C. Krothapalli and A. V. Deshmukh, "Design of negotiation protocols for multi-agent manufacturing systems," *International journal of production research,* vol. 37, pp. 1601-1624, 1999.
[70] H. V. D. Parunak, "Manufacturing experience with the contract net," *Distributed artificial intelligence,* vol. 1, pp. 285-310, 1987.
[71] T. Sandholm, "An implementation of the contract net protocol based on marginal cost calculations," in *AAAI*, 1993, pp. 256-262.
[72] M. d'Inverno and M. Luck, "Formalising the contract net as a goal-directed system," in *Agents breaking away*, ed: Springer, 1996, pp. 72-85.

[73] R. Davis and R. G. Smith, "Negotiation as a metaphor for distributed problem solving," *Artificial intelligence,* vol. 20, pp. 63-109, 1983.

[74] CloudSim.org. (Mar. 17). *VM Allocation Policy Abstract_Class*. Available: http://www.cloudbus.org/cloudsim/doc/api/org/cloudbus/cloudsim/VmAllocationPolicy.html

[75] S. Martello and P. Toth, *Knapsack problems: algorithms and computer implementations*: John Wiley & Sons, Inc., 1990.

Appendix: Publications/Presentations

Conferences

A. Al-Ou'n, M. Kiran, and D. D. Kouvatsos, "Using Agent-Based VM Placement Policy," in Future Internet of Things and Cloud (FiCloud), 2015 3rd International Conference on, 2015, pp. 272-281.

M. Kiran, K. Maiyama, H. Mir, B. Mohammad, and A. Al Oun, "Agent-Based Modelling as a Service on Amazon EC2: Opportunities and Challenges," in Utility and Cloud Computing (UCC), 2015 IEEE/ACM 8th International Conference on, 2015, pp. 251-255.

Workshops

A. Al-Ou'n, M. Kiran, and D. D. Kouvatsos, "The Advantages of Using the Multi-Agent System through Cloud Datacenter; Especially for the VM Allocation Process," Research Workshop NetPSEn 2015 on 'Networks and Performance-Security Engineering', Monday 21st Dec. '15.

Seminars

A. Al-Ou'n, M. Kiran, and D. D. Kouvatsos, "The Implementation and Validation of Agent-Based VM Placement Policy in Cloud Datacenter by Using the CloudSim Toolkit," University of Bradford ACM 'Association for Computing Machinery' Seminar, Bradford, Thursday 20th August 2015.

Journals

A. Al-Ou'n, M. Kiran, and D. D. Kouvatsos, In Preparation an Extensional Version of the Conference Paper 'Using Agent-Based VM Placement Policy' to Submit Soon to Future Generation Computer Systems Journal, https://www.journals.elsevier.com/future-generation-computer-systems.